VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA
EKONOMICKÁ FAKULTA

KATEDRA INFORMATIKY

Implementace technologie .NET v malé softwarové firmě

Implementation of the Technology .NET in a Small Software Company

Student: Bc. Martin Hatlapatka
Vedoucí diplomové práce: Ing. Vítězslav Novák, Ph.D.

Ostrava 2011

„Místopřísežně prohlašuji, že jsem celou práci, včetně všech příloh, vypracoval samostatně“.

V Ostravě dne 29.04.2011

……

podpis

Poděkování:

Děkuji za spolupráci panu Ing. Vítězslavu Novákovi, Ph.D. a panu Ing. Michalu Dolanskému za odborné vedení práce a jejich cenné rady.

Obsah

1. Úvod

Předmětem diplomové práce je vytvoření webové aplikace, která bude prezentovat zadavatelskou firmu (KS-program, spol. s.r.o.), její produkty, služby, a tyto produkty nabízet prostřednictvím jednoduchého elektronického obchodu. Webová aplikace bude založena na technologii .NET a bude ji možno spravovat prostřednictvím administrátorského rozhraní.

V dnešní době existuje široký výběr softwaru orientovaných na tvorbu webového obsahu. Od rozsáhlých a bohatých IDE na jedné straně až po malé a levné programy s nepříliš bohatou funkcionalitou. Každý nástroj pro tvorbu webového obsahu je primárně orientován na určitý typ uživatelů. Existují nástroje, se kterými mohou vytvořit hezké a dokonale funkční webové stránky i naprostí laici. Na opačném konci spektra jsou pak prostředky určené pro programátory s hlubokými a rozsáhlými znalostmi programovacích jazyků, webových standardů i samotných nástrojů, se kterými při tvorbě a správě webového obsahu pracují.

Do této skupiny můžeme zařadit například Visual Web Developer od společnosti Microsoft, který se také stal nástrojem pro tvorbu webové aplikace. Konkrétně Visual Web Developer 2010 Express. Microsoft Visual Web Developer 2010 je integrované vývojové prostředí, které se nepoužívá pro tvorbu webových stránek, ale webových aplikací. Používá ASP.NET pro rychlé a snadné sestavení požadovaných webových aplikací. Pomocným nástrojem přitom může být sada Microsoft Web Application Toolkits. Jedná se o řadu nástrojů pro vytvoření webových aplikací nebo aplikačních rozhraní na různá specifická témata, např. Social Network API, Web App Toolkit for Calendars, Web App Toolkit for Mobile Applications a další.

Dále umožňuje použít všechny moderní prostředky, které tvorbu webového obsahu umožňují nebo podporují. Jde o Ajax, Silverlight, PHP, Azure, DOM (Document Object Model), WCF (Windows Communication Foundation) a jiné. Samozřejmostí je spolupráce s databází, např. SQL Serverem.

Diplomová práce se skládá z několika částí. První část práce (Základní teorie a vymezení pojmů technologie .NET) se věnuje popisu základních metod

a technologií, které jsou v rámci vývoje webové aplikace využity. Tato kapitola vychází z větší části z cizích zdrojů. Druhá část (Analýza a popis stávajícího stavu) stručně informuje o stávajícím stavu ve společnosti KS-program, spol. s.r.o., dále se zabývá specifikací a analýzou požadavků webové aplikace. Ve třetí části práce (Návrh webové aplikace) je popsán postup vývoje, základní charakteristiky jednotlivých webových formulářů a vhled webové aplikace. Poslední část (Zhodnocení navrhovaného řešení) se zabývá hodnocením navrhovaného řešení a nástinem možných zlepšení či rozšíření.

2. Základní teorie a vymezení pojmů technologie .NET

V následující části práce uvádím a popisuji několik definic a pojmů, které považuji za důležité zmínit. Jednak proto, že se týkají dané problematiky a jednak z důvodů pochopení významu těchto pojmů, se kterými v diplomové práci pracuji. Dále pak uvádím stručné popisy využívaných technologií a krátké seznámení se s použitým vý-vojovým prostředím. Nejedná se o žádné podrobné definice jednotlivých termínů, ale pouze o základní principy a shrnutí významu daných technologií či informací.

2.1 Webová aplikace

Protože tématem diplomové práce je návrh webové aplikace založené na technologii .NET bylo by vhodné zmínit, co to webová aplikace je, v jakém smyslu bude s tímto pojmem nakládáno a co by taková webová aplikace měla splňovat.

Hlavním důvodem oblíbenosti webových aplikací je to, že je můžeme aktualizovat a spravovat bez nutnosti šířit a instalovat software na potenciálně tisíce uživatelských počítačů. Webové aplikace generují dynamicky sérii webových stránek ve standardním formátu HTML/XHTML, který je podporován běžnými prohlížeči. Každá webová stránka je prohlížeči dodána jako statický dokument. Sled těchto stránek pak vyvolává pocit interaktivity, např. reakce na vstup uživatele do formulářových prvků.

Na internetu, v literatuře a dalších zdrojích se můžeme setkat se spoustou definic webové aplikace. Z tohoto množství jsem vybral následující dvě definice:

„An application in which all or some parts of the software are downloaded from the Web each time it is run. It may refer to browser-based applications that run within the user's Web browser or to rich client applications that resemble local applications“
viz [10].

„Webová aplikace je aplikace poskytovaná uživatelům z webového serveru přes počítačovou síť internet, nebo její vnitropodnikovou obdobu (intranet). Webové aplikace jsou populární především pro všudypřítomnost webového prohlížeče jako klienta. Ten se pak nazývá tenkým klientem, neboť sám o sobě logiku aplikace nezná“ viz [11].

Webová aplikace bude splňovat následující obecné předpoklady vycházející z předešlých definic:

- Je založena na modelu klient/server.
- Pro komunikaci mezi klientem a serverem využívá protokolu HTTP.
- Rozhraním pro koncového uživatele je webový prohlížeč.
- Obsahuje formuláře pro odesílání informací uživatelům.

2.2 ASP.NET

Webové aplikace je možné vytvářet pomocí různých technologií. Jelikož firma, pro kterou webovou aplikaci vytvářím, již tuto technologii využívá, stala se tato technologie pro mě i pro firmu nejschůdnější alternativou. ASP.NET je nástupcem starší technologie aktivních serverových stránek (Active Server Pages). Nejedná se však o vylepšení původního ASP, ale o zcela novou technologii, díky začlenění ASP.NET v .NET Frameworku.

ASP.NET pracuje jiným způsobem než tradiční skriptovací technologie, mezi které patří klasické ASP nebo PHP. V porovnání s dřívějšími platformami webového vývoje se odlišuje v následujících skutečnostech:

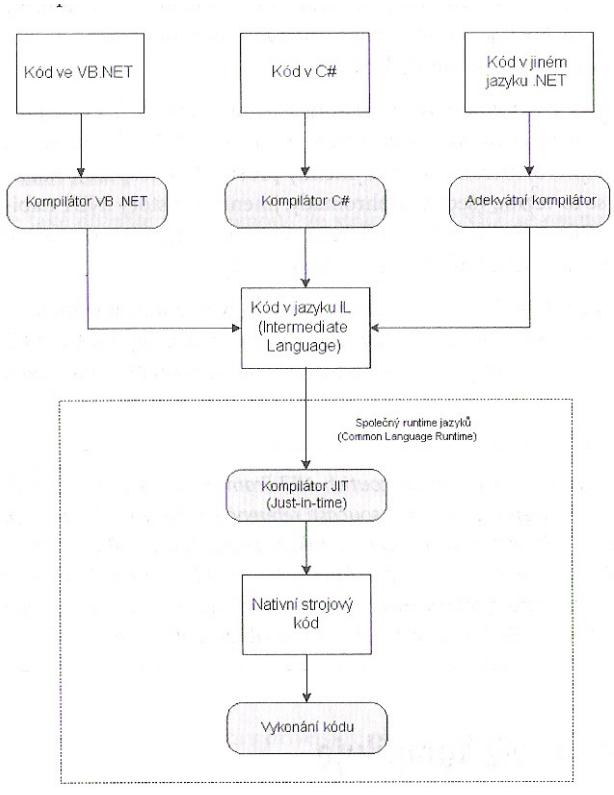
1) ASP.NET je integrováno s .NET Frameworkem.
.NET Framework je rozčleněn do pečlivě propracované kolekce funkčních částí, zahrnující více než 10000 typů. Každá třída v .NET Frameworku je seskupena do logického, hierarchického kontejneru, kterému se říká jmenný prostor (namespace). Různé jmenné prostory poskytují různé funkce. Jmenné prostory .NET nabízejí funkcionalitu téměř jakéhokoliv aspektu distribuovaného vývoje, od front zpráv (message queuing) až po otázky bezpečnosti. Této obrovské skupině se říká knihovna tříd (class library). [4]

2) ASP.NET se neinterpretuje, ale kompiluje.
Jedním z hlavních důvodů degradace výkonu ve skriptech ASP je to, že v kódu webových stránek s ASP se používají interpretované skriptovací jazyky. To znamená, že když se aplikace vykonává, musí skriptovací hostitel na serveru

interpretovat kód a přeložit jej do strojového kódu nižší úrovně. Tento proces je velmi pomalý.

Aplikace ASP.NET procházejí dvěma kompilačními etapami. V první etapě se kód, který jsme napsali, zkompiluje do přechodného jazyka, jenž se nazývá Microsoft Intermediate Language (MSIL), nebo prostě IL. Tento krok má za následek to, že v.NET je možné používat různé programovací jazyky. První kompilační krok může nastat automaticky, když se stránka poprvé požaduje, nebo se může vykonat předem (precompiling). Zkompilovanému souboru s kódem IL se říká assembly. [4]

Druhá úroveň kompilace nastává těsně předtím, než se stránka skutečně vykoná. V tomto okamžiku se kód IL zkompiluje do nativního nízkoúrovňového strojového kódu. Tato etapa se nazývá kompilace just-in-time (JIT) a probíhá stejně pro všechny aplikace .NET. [4] Oba kroky kompilačního procesu můžete vidět na obrázku 2.1.



Obrázek 2.1 Kompilační proces
zdroj: [4]

3) ASP.NET je vícejazyčné.

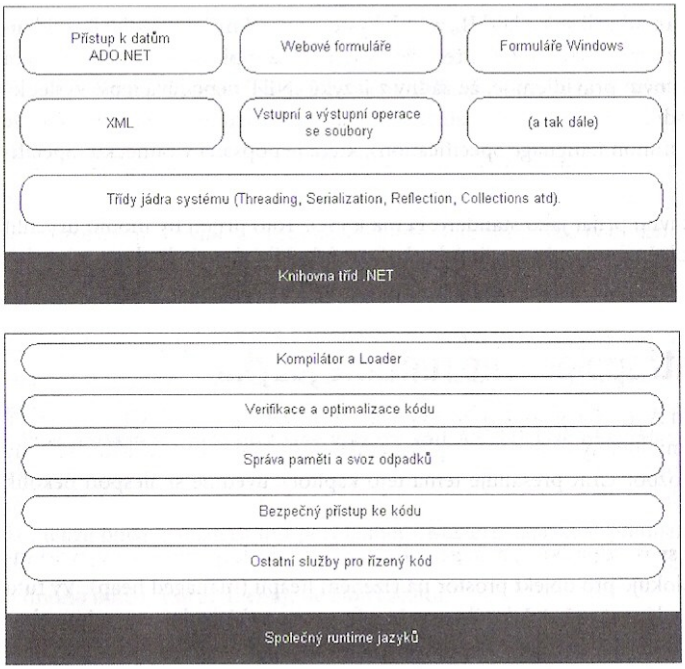
ASP.NET dává možnost psát kód v kterémkoliv z podporovaných jazyků .NET (mezi ně patří Visual Basic, C#, J# a mnoho dalších jazyků, které mají kompilátory od

jiných výrobců). Je to proto, že ať použijeme kterýkoliv jazyk, kód se vždy zkompiluje do IL. IL je odrazovým můstkem každé řízené aplikace (řízená aplikace, je jakákoliv aplikace, která byla napsána pro .NET a vykonává se uvnitř řízeného prostředí CLR (common language runtime)). V jistém smyslu je jazykem .NET právě IL. Je to jediný jazyk, kterému CLR rozumí. [4]

4) ASP.NET běží uvnitř společného runtime jazyků.
Patrně nejdůležitějším aspektem enginu ASP.NET je to, že běží uvnitř runtimového prostředí CLR. Některé výhody CLR: [4]

- **Automatická správa paměti a svoz odpadků (garbage collection)**. Vždy, když aplikace vytvoří instanci nějakého objektu, CLR alokuje pro objekt prostor na řízeném heapu (managed heap). Uvnitř CLR v pravidelných intervalech jezdí garbage collector, který automaticky požaduje navrácení nepoužívané paměti objektů, k nimž už nelze přistupovat.
- **Typová bezpečnost**. Když kompilujeme nějakou aplikaci, přidá .NET do assembly informaci specifikující některé podrobnosti o ní. Například dostupné třídy, jejich členy, jejich datové typy atd. výsledkem je, že assembly zkompilovaného kódu budou plně soběstačné. Jiní lidé je budou používat, aniž by se požadovaly nějaké podpůrné soubory. Tato vrstva také zamezuje vzniku různých nízkoúrovňových chyb.
- **Rozšířitelná metadata**. Informace o třídách a členech jsou pouze jedním z typů metadat, která .NET ukládá do zkompilované assembly. Metadata popisují kód a umožňují poskytnout dostatečné informace pro runtime nebo pro jiné služby. Metadata mohou například sdělit debuggeru, jak má trasovat kód, nebo mohou sdělit Visual Studiu, jak má zobrazit nějaký vlastní ovládací prvek v návrhovém režimu.
- **Strukturované zpracování chyb**. Jazyky .NET nabízejí strukturované zpracování výjimek, které umožňuje uspořádat kód, jímž budeme reagovat na chyby, logicky, stručně a výstižně.
- **Multithreading**. CLR poskytuje fond vláken (pool of threads), který mohou využívat různé třídy. Můžeme například volat metody, číst soubory, nebo komunikovat s webovými službami asynchronně, aniž bychom museli explicitně vytvářet nová vlákna.

Vysokoúrovňový pohled na CLR a .NET Framework předvádí obrázek 2.2.



Obrázek 2.2 CLR a .NET Framework
zdroj: [4]

5) ASP.NET je objektově orientované.

Klasické ASP poskytuje slabší objektový model. Poskytuje pouze malou sadu objektů a v podstatě tvoří pouze tenkou vrstvu nad zdrojovými detaily HTTP a HTML. Oproti tomu je ASP.NET opravdu objektově orientované. Nejenom že kód má úplný přístup ke všem objektům v .NET Frameworku, ale také můžeme těžit z veškerých konvencí objektově orientovaného prostředí (OOP). Můžeme například vytvářet opětovně využitelné třídy, standardizovat kód rozhraními, rozšiřovat existující třídy děděním, nebo začlenit nějakou užitečnou funkcionalitu do zkompilované komponenty určené k dalšímu šíření. [4]

6) ASP.NET podporuje různá zařízení a různé prohlížeče.

Jednou z největších výzev, které čelí weboví vývojáři, je značná různorodost prohlížečů, které je potřeba podporovat. Všelijaké prohlížeče, jejich verze a jejich konfigurace podporují standard HTML různě. Vývojáři musejí zvolit, zda je potřeba obsah realizovat podle HTML 3.2, 4.0 nebo podle XHTML 1.0 či WML (Wireless Markup Langugage), což je určeno pro mobilní zařízení. [4]

ASP.NET se vypořádává s tímto problémem velmi inteligentně. Disponuje bohatě vybavenou skupinu webových ovládacích prvků. Ty realizují svůj kód HTML adaptivně, což znamená, že berou v úvahu schopnosti klienta. Jako vhodný příklad zde mohou posloužit ovládací prvky pro ověřování platnosti vstupních dat (tzv. validátory), které pro posílení schopnosti svého chování používají JavaScript a DHTML (Dynamic HTML), za předpokladu, že je klient podporuje. Validačním ovládacím prvkům to umožňuje dynamicky zobrazovat chybové zprávy, aniž by uživatel musel stránku odesílat na sever k dalšímu zpracování. [4]

7) ASP.NET se snadno rozmisťuje a konfiguruje.
Každá instalace .NET Frameworku poskytuje stejné jádro tříd. Z toho plyne, že aplikace ASP.NET se rozmisťují relativně snadno. Ve většině případů stačí pouze zkopírovat všechny soubory do nějakého virtuálního adresáře na ostrém serveru. Pokud má hostitelský stroj .NET Framework, nejsou potřeba žádné registrační kroky, které by mohly být časově velmi náročné. [4]

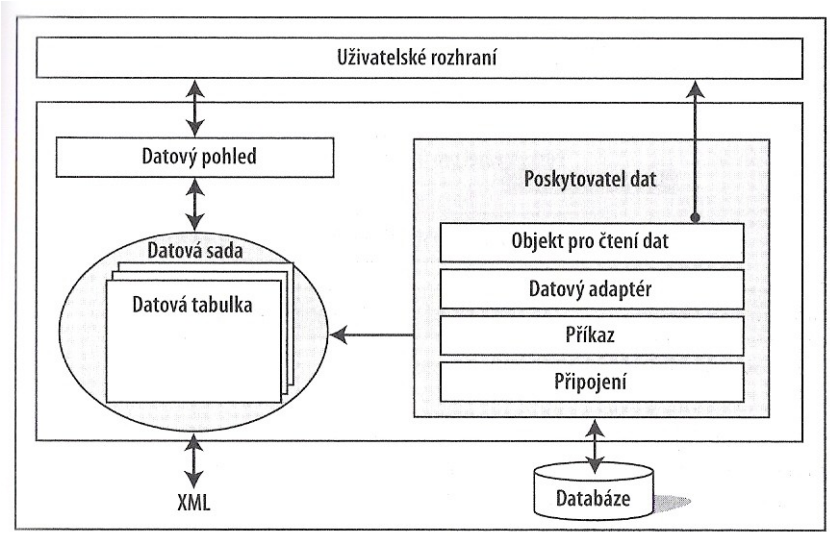
Distribuce komponent je také snadná. Když rozmísťujeme aplikaci, opět stačí jen zkopírovat všechny komponenty. Pokud umístíme komponenty na správné místo (do podadresáře Bin adresáře webové aplikace), bude je engine ASP.NET detekovat automaticky a učiní je dostupnými pro kód webové stránky. [4]

ASP.NET fázi konfigurace usnadňuje tím, že minimalizuje závislost na nastavení v IIS (Internet Information Services). Většina nastavení ASP.NET se ukládá do souboru web.config, který existuje právě pro tento účel. Soubor web.config se umísťuje do téhož adresáře, kde máme své webové stránky. Obsahuje hierarchická seskupení nastavení aplikace, která jsou uložená ve formátu XML. [4]

2.3 ADO.NET

Většinu aplikací nelze sestavovat bez toho, aby nějakým způsobem pracovaly s databází. Databáze totiž řeší problémy spojené se získáváním a uchováváním dat. Běžně softwarové aplikace pracují s jednou nebo více databázemi. Klientská část musí mít k dispozici nějaké mechanismy pro připojení k databázím, k čemuž slouží právě knihovna ADO.NET. Každá aplikace, která je postavena na platformě .NET

je tak na této knihovně závislá. Na obrázku 2.3. je znázorněna architektura knihovny ADO.NET.



Obrázek 2.3 Architektura knihovny ADO.NET
zdroj: [1]

S příchodem rozhraní .NET Framework 3.5 byla do Visual Studia zařazena nová verze knihovny ADO.NET: ADO.NET 3.5. Pokud tedy chceme pracovat s prvky této knihovny, musíme mít k dispozici rozhraní Entity Framework a nástroje Entity Framework Tools.

Rozhraní ADO.NET 3.5 Entity Framework umožňuje zaměřit se na data prostřednictvím objektového modelu, díky čemuž se snadněji abstrahuje logické datové schéma do konceptuálního modelu, s nímž je možné pracovat pomocí nového poskytovatele dat nazvaného EntityClient.

Prostřednictvím tohoto rozhraní (ADO.NET 3.5 EF), které jsem také ve webové aplikaci využil je možné psát méně kódu pro přístup k datům, snižují se nároky na údržbu a struktura dat je abstrahována do aplikačně přívětivější formy.
Model rozhraní ADO.NET 3.5 EF se skládá ze tří aktivních vrstev:

- konceptuální vrstva,
- mapovací vrstva,
- logická vrstva.

Tyto tři vrstvy umožňují mapování dat z relační databáze do objektově orientovanějšího aplikačního modelu. Vrstvy se definují pomocí souborů XML, takže

místo tradičního relačního datového modelu můžeme využívat objektově orientovaný konceptuální model.

Řešení entitním datovým modelem rozhraní ADO.NET 3.5 spočívá v tom, že vůbec není nutné v případě nějaké změny zasahovat do zdrojového kódu, ale úplně stačí zanést tyto změny ve schématu do mapovacího souboru XML.

[1]

2.4 LINQ

LINQ (Language Integrated Query) je sada jazykových rozšíření, která umožňují vykonávat dotazy, aniž bychom se museli vzdát jazyka C#. LINQ definuje klíčová slova, pomocí nichž se tvoří dotazovací výrazy. Tyto výrazy mohou filtrovat, vybírat, řadit, seskupovat a transformovat data. LINQ umožňuje používat stejné dotazovací výrazy nad různými zdroji dat. Mezi tyto zdroje můžeme zařadit:

- LINQ to Objects – umožňuje klást dotazy na kolekce objektů uložených v paměti.
- LINQ to DataSet – umožňuje klást dotazy na sadu dat uložených v paměti.
- LINQ to SQL – umožňuje se dotazovat na databázi SQL Serveru, aniž bychom museli psát kód pro přístup k datům.
- LINQ to XML – tento zdroj umožňuje přečíst soubor XML, aniž bychom museli používat specializované třídy .NET určené pro práci s XML.

Jednou z nejvíce používaných částí je ta část LINQ pojmenovaná jako LINQ to SQL. Její hlavní doménou je to, že překládá výrazy LINQ do dotazů SQL za scénou. Tyto dotazy jsou vykovávány, když potřebujeme data. Dále obsahuje mechanismus pro aktualizace. Poskytuje systém pro sledování změn pro všechna získaná data, čímž je možné modifikovat objekty, na které jsme se dotazovali a následně v jednom okamžiku potvrdit celou dávku změn.

LINQ to SQL neposkytuje žádnou funkcionalitu navíc, kterou bychom nemohli duplikovat s kódem ADO.NET, s LINQ to Objects nebo LINQ to DataSet. Ovšem dokáže v několika ohledech usnadnit práci:

- **Méně kódu.** Při dotazování do databáze není třeba psát kód ADO.NET.

- **Flexibilní dotazovací výbava.** Díky dotazovacímu modelu LINQ se nemusíme potýkat s příkazy SQL.
- **Sledování změn a dávkové aktualizace.** Bez psaní kódu ADO.NET je možné změnit velké množství podrobnosti v datech, na které jsme se dotazovali a následně v jedné dávce potvrdit aktualizace.

[4]

2.5 XHTML

XHTML (eXtensible HyperText Markup Language) je značkovací jazyk, který se využívá pro tvorbu hypertextových dokumentů v prostředí internetu a je založen na XML. Jazyk XHTML je následovníkem staršího HTML. Je založen na univerzálním
značkovacím jazyku SGML (Standard Generalized Markup Language) a využívá tagy ke strukturování textu.

Stránky XHTML musí splňovat mnohem přísnější syntaktická pravidla než dnešní HTML stránky. Dnešní prohlížeče se dokážou vypořádat s mnoha chybami ve stránkách, jako jsou překřížené elementy, špatně spárované tagy apod. XHTML však vyžaduje, aby všechny stránky byly XML-dokumenty, musí proto splňovat požadavky stejné jako každý XML-dokument. K Základnímu požadavku patří dodržení správné struktury (well-formedness). Mezi další požadavky můžeme zařadit například:

- Všechny elementy a atributy musí být zapisovány malými písmeny.
- Všechny tagy musí být párové a každý tag musí být uzavřen. Pro nový řádek nelze použít <br>, ale musí se použít uzavřený tag <br/> nebo párový tag
</br>.
- obsah atributů se musí uzavírat do uvozovek nebo apostrofů.
- Tagy se nesmí křížit. Např. První odstave Druhý odstave musíme přepsat do tvaru První odstavec Druhý odstavec .

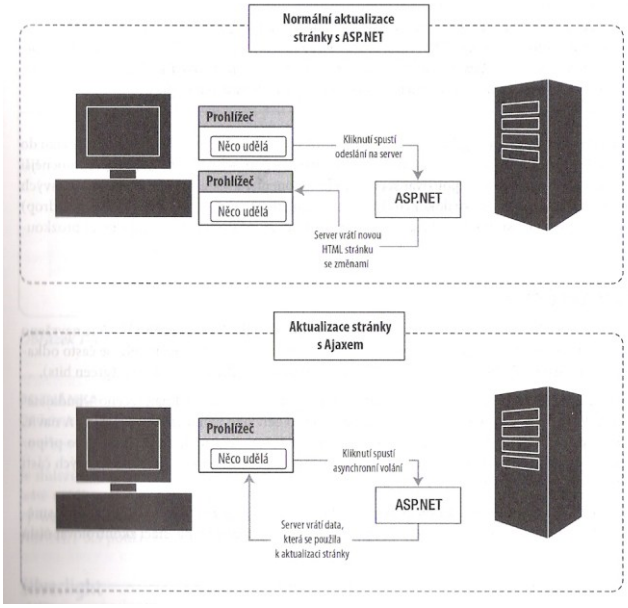
[3]

2.6 ASP.NET AJAX

Jednou z dalších technologií, využívanou v rámci webové aplikace je abstraktní vrstva s názvem ASP.NET AJAX, která rozšiřuje ASP.NET o působivé funkce ve stylu Ajaxu.

Pokud například potřebujeme aktualizovat informace pro některé části stránky, jediný způsob, jak se to dá udělat v obyčejném ASP.NET, je odeslat stránku na server a získat zpět úplně nový dokument HTML. Toto řešení sice funguje, ale rozhodně není nepřerušované. Řešením může být psaní více kódu pokročilejším způsobem na straně klienta.

Ajax je zkratka pro programátorskou techniku aplikovanou u klienta, která umožňuje, aby stránka zavolala server a aktualizovala svůj obsah, aniž by spustila kompletní odeslání zpět na server. Stránka s Ajaxem obvykle používá skript u klienta tak, že za scénou odpálí asynchronní požadavek. Server obdrží požadavek, vykoná nějaký kód a vrátí data, která stránka potřebuje. Kód klienta obdrží tato nová data a s jejich pomocí provede nějakou akci, například aktualizuje část stránky. Přestože je Ajax koncepčně velmi prostý, umožňuje vytvářet stránky, které fungují plynuleji, takže aplikace mohou běžet opravdu nepřerušovaně. Rozdíl ilustruje obrázek 2.4.



Obrázek 2.4 Obyčejné stránky na straně serveru vs. Ajax
zdroj: [4]

ASP.NET AJAX se skládá ze dvou klíčových částí – z části na straně klienta a z části na straně serveru. Klientská část je sada knihoven JavaScriptu. Tyto knihovny nejsou

žádným způsobem spojeny s ASP.NET. V praxi to znamená, že na webových stránkách je mohou používat i programátoři jiných jazyků než ASP.NET. Klientské knihovny toho nenabízí příliš mnoho, alespoň co se týče funkcionality. Místo toho vytvářejí základ, který je nezbytný pro vývoj stránek prostřednictvím ASP.NET AJAX. Serverová část ASP.NET AJAX funguje na vyšší úrovni. Obsahuje ovládací prvky a komponenty, které používají klientské knihovny JavaScriptu.

Přehled funkcí, které ASP.NET AJAX poskytuje:

- **Rozšíření jazyka JavaScript.** Tato rozšíření částečně přibližují JavaScript k moderním, objektově orientovaným, programovacím jazykům s podporou jmenných prostorů, rozhraní, dědičnosti a reflexe.
- **Vzdálené volání metod.** Umožňuje získat informace ze serveru, aniž bychom museli odesílat celou webovou stránku zpět na server. Pracuje se silně typovými metodami.
- **Služby ASP.NET.** Funkcionalita, která umožňuje zavolat server pro komunikaci s modelem ASP.NET. Pracuje se službami, které získávají data z aktuálního profilu uživatele nebo používají informace formulářové autentizace.
- **Obnovení části stránky.** Prostřednictvím prvku UpdatePanel je možné aktualizovat části stránek bez toho, aby bylo nutné posílat na server celou stránku. Pro řízení procesu aktualizace není třeba psát žádný kód JavaScriptu.
- **Předem vytvořené ovládací prvky.** ASP.NET AJAX Control Toolkit obsahuje přes 30 ovládacích prvků, které používají funkcionalitu ASP.NET AJAX pro dosažení dobře vypadajících efektů.

[4]

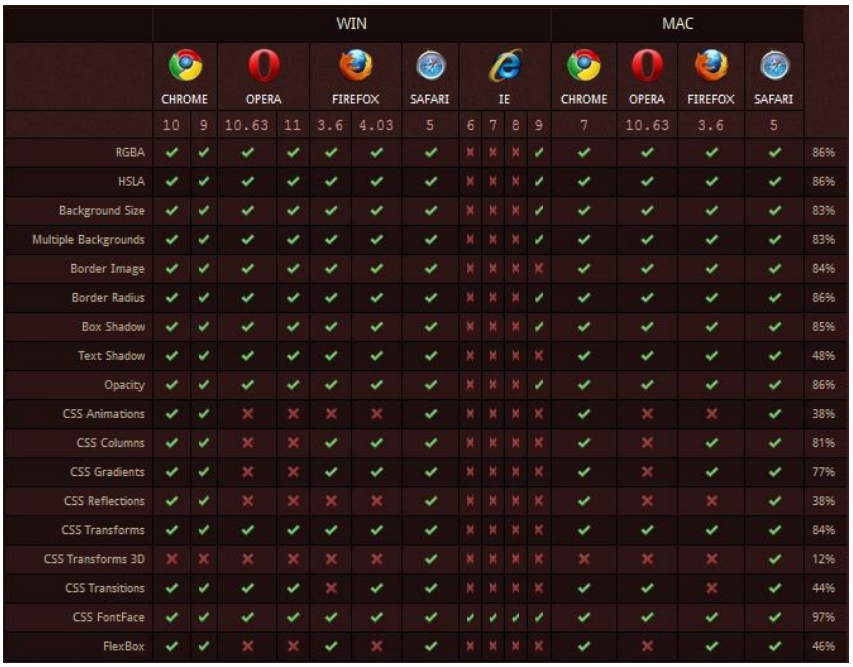
2.7 CSS

Je kolekce metod pro grafickou úpravu webových stránek. Zkratka znamená Cascading Style Sheet, česky „kaskádové styly“. Kaskádové, protože se na sebe mohou vrstvit definice stylu. CSS nabízí rozsáhlejší formátovací možnosti než samotné HTML. Nejlépe je to vidět při vytváření stylů bloků textů.

Jedním z nejvýznamnějších přínosů je oddělení struktury stránky od stylu. Jazyk HTML je především určen pro popis struktury hypertextového obsahu. Tentýž obsah je možné prezentovat v různých podobách. Díky CSS lze dosáhnout menší datové velikosti stránek a tím i k rychlejšímu načítání. [4]

Ve webové aplikaci jsou využity některé prvky CSS3. Jejich přínosem je zejména zvýšení vizuální přívětivosti webu. Použity jsou např. stíny, barevné přechody (gradients) nebo popisky obrázků (Fading Image Captions).

Malou nevýhodou těchto efektů je to, že jejich vykreslování nepodporují všechny webové prohlížeče. Asi nejhůř na tom je IE (Internet Explorer). Sice je stále nejpoužívanějším prohlížečem, ale jeho statistiky dlouhodobě klesají. Daleko lépe je na tom u nás hojně používaný Firefox nebo Google Chrome. Případná nefunkčnost těchto efektů nemá v žádném případě negativní dopad na funkčnost webové aplikace. Na následujícím obrázku 2.5 jsou v přehledné tabulce zobrazeny podporované efekty v jednotlivých prohlížečích.



Obrázek 2.5 CSS3 vlastnosti
zdroj: [8]

2.8 JavaScript

JavaScript je skriptovací jazyk, který se používá na webových stránkách pro tvorbu programů, nebo pro oživení stránek. Pomocí JavaScriptu a CSS (DHTML) můžeme dosáhnout zajímavých dynamických efektů. JavaScript se oproti PHP skriptům vykonává přímo v prohlížeči čtenáře, proto je rychlý a k jeho funkčnosti je zapotřebí jen internetový prohlížeč, který podporuje JavaScript.

Pomocí JavaScriptu lze vytvářet programy, které se spustí po určité události a měnit stránku i po načtení. Spolu s CSS lze dosáhnout dynamických efektů.

Skript je možné do stránky vložit několika způsoby:

- Jako externí soubor s příponou js.
- Přidáním značky <script>, která obsahuje javascriptový kód. Tento kód je možné spustit automaticky při načtení stránky nebo pomocí funkce JavaScriptu, která bude zavolána jako reakce na událost na straně klienta.
- Vložením kódu přímo do atributu nějakého HTML prvku. Tento způsob je nejvhodnější pro kód menšího rozsahu.

[7]

2.9 UML

UML (Unified Modeling Language) je modelovací jazyk, který se v softwarovém inženýrství využívá pro vizualizaci, specifikaci, návrh a dokumentaci programových systémů. [2]

Jazyk UML je souhrnem převážně grafických notací k vyjádření analytických a návrhových modelů. UML umožňuje modelovat jednoduché i složité aplikace pomocí stejné formální syntaxe, a proto je možné sdílet výsledky práce s ostatními návrháři.
Model v UML se skládá z různých diagramů, jež představují průhledy na různé části sémantického základu navrhované aplikace. Jazyk UML rozeznává pět základních pohledů na systém:

- pohled případů užití,
- logický pohled,

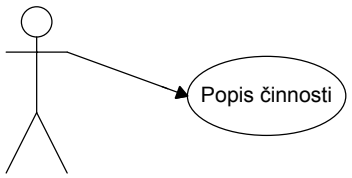
- procesní pohled,
- implementační pohled,
- pohled nasazení.

Pohledy jsou pak konkretizovány v několika typech diagramů, z nichž jsou v rámci diplomové práce využity zejména diagramy případů užití. [6]

2.9.1 Use Case diagram

Zachycuje přesně funkčnost, která bude budoucím informačním systémem pokryta a vymezuje tak jednoznačně rozsah prací. Každý Use Case, popisuje jeden ze způsobu použití systému. Popisuje jednu jeho požadovanou funkčnost.

Na obrázku je znázorněný jednoduchý Use Case diagram. Postava představuje aktéra, což je role, ve které vystupuje uživatel v rámci komunikace se systémem, elipsa představuje případ užití a spojnice představuje interakci mezi uživatelem a případem užití.



Kromě vztahů mezi aktéry a případy užití můžeme také definovat několik vztahů mezi případy užití samotnými.

V diplomové práci jsou používány následující dva vztahy:

- Relace <<include>> vyčleňuje stejné chování ze dvou a více případů užití do samostatného případu užití. Základní případ užití není soběstačný.
- Relace <<extend>> přidává k základnímu případu užití nové, rozšiřující chování. Základní případ užití je zcela soběstačný.

[2]

2.9.2 Případ užití

Je to sada scénářů, které spojuje dohromady společný cíl. Detailně popisují jednotlivé části systému pomocí tabulkově-textové specifikace. [2] V diplomové práci se můžete setkat s následujícím formátem:

Název případu užití	činnost
Identifikace případu užití	UC0
Aktéři	uživatel
Vstupní podmínky	-
Kroky případu užití	1. První krok 2. Další kroky …
Výstupní podmínky	-

2.9.3 ER-diagram

Tento model popisuje objekty a jejich vztahy buď textovým zápisem nebo pomocí ER diagramů (Entity Relationship Diagrams). Používá kombinace textových formálních zápisů, grafického zobrazení typů entit, atributů, vztahů mezi entitami a doplňujících textových informací. Model ER má podobný význam jako vývojový diagram pro návrh algoritmu. Je určen pro návrh struktury databáze, pro popis hlavních vlastností dat. [6]

2.10 Visual Studio

Visual Studio je dostupné v několika verzích. Pro tvorbu webové aplikace jsem použil Visual Web Developer 2010 Express Edition, která je volně dostupná ke stažení na stránkách výrobce.

Visual Studio je profesionální vývojářský nástroj, který podporuje komplexní soustavu různých designérských nástrojů, včetně sady nástrojů pro ladění, IntelliSense, jež při psaní kódu zachytává chyby a nabízí různá doporučení. Visual studio také podporuje robustní model kódu v pozadí (code-behind), kdy se kód .NET, který píšeme, odděluje od značek webové stránky. Visual Studio má také zabudovaný server, který je určený pro testování, takže vytvořené weby se ladí snadno a bezpečně.

Mezi přednosti Visual Studia patří:

- **Integrovaný webový server**. Do Visual studia je integrován testovací webový server, díky němuž můžeme spouštět weby přímo z prostředí vývojářského nástroje. Je to také lepší z hlediska bezpečnosti, protože si můžeme být jisti, že žádný externí počítač nemůže spustit náš aktuálně testovaný web. Testovací server totiž akceptuje jen připojení z místního počítače.
- **Vývoj ve více jazycích**. Visual studio umožňuje psát kód buď v jednom jazyku, nebo v několika jazycích, a přitom používat stále stejné rozhraní (IDE). Navíc Visual Studio umožňuje vytvářet jednotlivé webové stránky v různých jazycích a zařadit je všechny do téže webové aplikace.
- **Intuitivní styl psaní kódu**. Visual studio nám pomáhá při psaní kódu tím, že jej automaticky formátuje, odsazuje a používá různé barvy pro odlišení různých částí kódu, jako jsou třeba komentáře.
- **Rychlejší vývoj**. Funkce týkající se pohodlí umožňují pracovat rychle a efektivně. Patří mezi ně IntelliSense (označuje chyby a doporučuje opravy), výkonné vyhledávání a nahrazování, či automatické převádění kódu na komentáře.
- **Ladění**. Visual Studio obsahuje nástroje určené pro ladění. S jejich pomocí můžeme snadno vystopovat chyby a diagnostikovat podivné chování. Svůj kód můžeme vykonávat řádek po řádku, nastavovat inteligentní body přerušení (breakpoints). Můžeme také prohlížet informace nacházející se v paměti počítače.

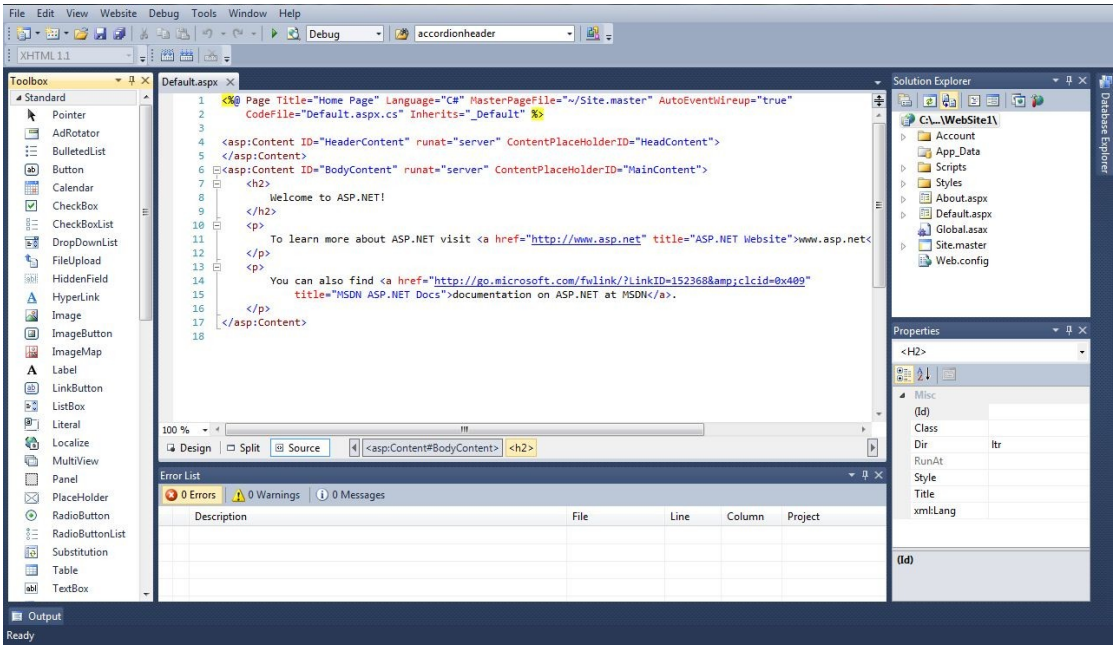
Integrované vývojové rozhraní Visual Studia je rozděleno do několika částí, mezi které patří:

- Solution Explorer – obsahuje výpis souborů a složek nacházející se ve složce webové apliace.
- Toolbox – zobrazuje zabudované serverové ovládací prvky ASP.NET a další ovládací prvky od jiných výrobců, které je možné do Toolboxu přidat.
- Server Explorer – umožňuje přístup k databázím, systémovým službám, frontám zpráv a jiným zdrojům umístěným na straně serveru.

- Properties – v tomto okně je možné konfigurovat aktuálně vybraný prvek, ať už je to soubor v průzkumníkovi řešení nebo ovládací prvek na návrhové ploše webového formuláře.
- Error List – v tomto okně Visual Studio oznamuje detekované chyby v kódu a chyby, které nejsou dosud opraveny.
- Task List – vypisuje komentáře začínající předdefinovanou kombinací znaků. To nám dává průběžný přehled o těch částech kódu, které plánujeme změnit.
- Document – umožňuje vytvářet webovou stánku stylem drag-and-drop a editovat kód v souborech, které máme uvnitř Solution Exploreru.
- Class View – zobrazuje odlišný pohled na aplikaci.
- Manage Styles and Apply Styles – umožňuje modifikovat styly v propojeném stylovém předpisu a aplikovat je na aktuální webovou stránku.

[4]

Vývojové rozhraní Visual Studia můžete vidět na obrázku 2.6.



Obrázek 2.6 Rozhraní Visual Studia
zdroj: [4]

2.11 Charakteristika KS-Program

Jelikož webovou aplikaci vytvářím pro firmu, proto v následující části diplomové práce uvádím její základní informace.

Název subjektu: KS - program, spol. s r.o.

IČO: 43963617

Sídlo: Vsetín, Rokytnice 413, PSČ 75501

Předmět: poskytování software a poradenství v oblasti hardware a software

2.10.1 Historie společnosti

„Společnost KS - program vznikla již v roce 1991. Původní organizace Kancelářské stroje, s.p., pobočka Vsetín se po roce 1989 rozdělila na více samostatných společností a jednou z nich byla právě společnost KS - program. V době jejího vzniku se zaměřila především na oblast vývoje mzdových systémů pro osobní počítače, které v té době začínaly nahrazovat velké sálové počítače“.

„Hlavním cílem nově založené společnosti byl vývoj a implementace vlastního mzdového informačního systému. Klíčovými zákazníky měly být především střední a velké soukromé společnosti z výrobního sektoru a významní poskytovatelé služeb. Vývoj mzdového systému byl zahájen ve vývojovém prostředí Informix-4GL pro operační systém UNIX a databázovou platformu INFORMIX. Dynamický vývoj v oblasti IT a zejména masivní nástupu operačního systému Microsoft Windows vedl velmi záhy k rozhodnutí vedení společnosti investovat do vývoje komplexního personálního a mzdového informačního systému v novém vývojovém prostředí podporujícím práci klienta v prostředí MS Windows“.

„Tento vývoj byl zahájen v roce 1995. Pro nové vývojové prostředí byl zvolen vývojový nástroj PowerBuilder společnosti PowerSoft později Sybase Inc. Přechod na tento nový vývojový nástroj umožnil mimo podstatnou změnu klientského prostředí i velmi podstatné rozšíření portace aplikace na více databázových platforem. Tím se podstatně zvýšila možnost volby databáze samotným koncovým uživatelem. První fáze vývoje byla dokončena v roce 1997 a první zákazníci tento systém používají od roku 1998. Přesto až do roku 2004 udržovala firma KS - program souběžně dva systémy jak pro znakové, tak i pro grafické prostředí. Poslední zákazník byl převeden na grafickou verzi aplikace k 01. 01. 2005“.

„Současně se od roku 1995 původní mzdový systém začal rozšiřovat o moduly pro podporu řízení lidských zdrojů. Začal tak vznikat komplexní personální a mzdový

informační systém. Dalším významným mezníkem ve vývoji společnosti bylo zahájení spolupráce s řadou obchodních partnerů. Především v roce 2000 navázala společnost KS - program spolupráci s řadou obchodních partnerů především z oblasti dodavatelů celopodnikových informačních systémů ERP. Mezi první patřil systém MAX a SAP. Současně se začala rozvíjet velmi intenzivní spolupráce s významnými dodavateli docházkových a stravovacích systémů“.

„Intenzivní rozšíření rychlého internetu vedlo v roce 2004 k rozhodnutí začít využívat toto komunikační médium k rozsáhlejšímu a přístupnějšímu řízení lidských zdrojů. Byl proto zahájen vývoj aplikace KS portál, která by umožňovala přístup k datům odkudkoliv a kdykoliv. Původní verze programovaná v jazyku html byla nahrazena v roce 2005 vývojovým nástrojem Microsoft Visual studio“.

„Dnes tedy nabízí společnost KS-program personální a mzdový informační systém pro zákazníky z České i Slovenské republiky, případně i personální informační systém pro všechny země Evropské unie“.

[9]

2.11.2 Poskytované služby

Implementační služby

Implementace personálního informačního systému je ucelený soubor služeb určený k plynulému nasazení personálního a mzdového informačního systému do plného provozu v co nejkratším čase. Součástí implementace je převod dat z původního systému, instalace, interface na okolní systémy, školení uživatelů i autorský dozor při ostrém provozu.

Implementace a její rozsah vždy závisí na požadavcích zákazníka směrem k systému. Dle definovaných požadavků je navržen postup implementace v jednotlivých krocích tak, aby v určeném termínu, byl systém provozuschopný a přinášel konkrétní výsledky.

Služby technické podpory

Technická podpora je nedílnou součástí budoucího fungování systému a současně spokojenosti uživatelů. V rámci technické podpory jsou zákazníkům poskytovány následující služby:

- Zajištění legislativní správnosti aplikací dle české a slovenské legislativy, písemné upozorňování na změnu legislativy.
- Písemné upozorňování na změnu metodiky.
- Zasílání oznámení o všech změnách v aplikacích.
- Dodávka nových verzí - update, upgrade.
- Možnost autorského dozoru při zpracování mezd.
- Metodika organizace zpracování mezd pomocí aplikace KS mzdy.
- Metodická podpora řízení lidských zdrojů.
- Podpora při stanovení kompetencí, nastavení základních přístupových práv.
- Možnost provedení servisních zásahů u zákazníka pomocí vzdáleného přístupu.
- Informativní semináře o změnách v legislativě a aplikaci KS mzdy.
- Informační buletin k aplikaci KS mzdy a KS personalistika.

[9]

3. Analýza a popis stávajícího stavu

Společnost KS-Program, pro kterou je webová aplikace určena již provozuje firemní web. Tento web je vytvořen dodavatelskou firmou, která také zabezpečuje její provoz. Web prezentuje společnost KS-Program, její činnost a její produkty. Společnost by chtěla provozovat nový firemní web, založený na technologii ASP.NET. Tuto technologii již firma využívá při vývoji jiných aplikací, mohla by tak sama nový firemní web spravovat a rozšiřovat.

Webová aplikace by měla splňovat následující požadavky:

- Prezentovat společnost KS-Program.
- Prezentovat produkty společnosti.
- Prezentovat služby společnost.
- Součást aplikace, by měl být elektronický obchod s nabídkou produktů společnosti.
- Administrátorské rozhraní pro správu aplikace.

3.1 Formální specifikace

Pod pojmem požadavek bude v následujícím textu rozuměno popis (specifikaci) jisté funkce nebo vlastnosti, která by měla být ve vyvíjené webové aplikaci implementována.

Rozlišujeme dva základní typy požadavků:

- funkční – specifikují požadavky na funkčnost webové aplikace,
- nefunkční – specifikují jisté vlastnosti webové aplikace, případně podmínky omezující fungování webové aplikace.

3.1.1 Funkční požadavky

Správa uživatelů

V systému vystupují 3 role uživatelů. Pro každou roli jsou definována práva a funkce, které daný uživatel může využívat.

- Neregistrovaný uživatel – jedná se o uživatele, který se při vstupu na stránky nepřihlásí. Tento uživatel má omezené funkce (nebude moci nakupovat).
- Registrovaný uživatel – uživatel, který je v aplikaci zaregistrovaný a přihlášený. Tento uživatel má možnost využívat veškeré funkce systému, mimo funkce určené pro administrátora aplikace.
- Administrátor – administrátor zabezpečuje správu systému. Má stejná práva jako registrovaný uživatel plus speciální práva pro správu webové aplikace.

Správa uživatelů obsahuje následující funkce:

- Registrace – uživatel, který ještě nemá vytvořený účet, se nejprve musí registrovat. Registrace vyžaduje vyplnění následujících údajů (uživatelské jméno, e-mail, heslo, potvrzení hesla).
- Přihlášení – po registraci má možnost uživatel se přihlásit. Pokud tak neučiní, nebude moci vstoupit do košíku.
- Odhlášení - odhlášení ze systému.

- Správa vlastního účtu – každý přihlášený uživatel má možnost editovat vlastní údaje jako jsou změna hesla, změna e-mailu nebo vyžádání si zapomenutého hesla.

Hodnocení modulů

Každý přihlášený uživatel může hodnotit jednotlivé moduly případně kompletní programový balík. Rozsah hodnotící stupnice je od jedné do pěti. Kde číslo pět je nejvyšší hodnocení. K hodnocení může být přidán komentář.

Nakupování modulů

Webová aplikace zahrnuje jednoduchý elektronický obchod, který umožňuje registrovaným, respektive přihlášeným uživatelům vytvořit objednávku na konkrétní modul nebo kompletní softwarový balík prostřednictvím nákupního košíku.

Následující požadavky se týkají role administrátora.

1. Správa produktů:
    - vložení produktu,
    - odstranění produktu,
    - editace produktu.
2. Správa modulů:
    - vložení modulu k danému produktu,
    - odstranění modulu,
    - editace modulu.
3. Hodnocení modulů:
    - odstranění hodnocení.
4. Přehled objednávek:
    - editace objednávky,
    - odstranění objednávky,
    - podrobnosti objednávky - přehled o tom, který uživatel objednávku učinil, jaké zboží si objednal, celkovou cenu, datum vystavení atd.
5. Správa uživatelských účtů:
    - editace uživatele,
    - odstranění uživatele,

- schvalování účtu,
- odemknutí účtu,
- přidání a odstranění uživatele do příslušné role.

3.1.2 Nefunkční požadavky

Implementace

Webová aplikace je implementována pomocí platformy ASP.NET C#.

Data

Správu dat zajišťuje databázový systém MS SQL.

Výstup

Výstup webové aplikace je ve formátu XHTML a formátován pomocí CSS tak, aby se správně zobrazil v těchto prohlížečích (Mozilla Firefox 3.6, 4; Google Chrome; Internet Explorer 8, 9).

3.2 Specifikace uživatelského vzhledu

- Vytvoření webových stránek, které budou reprezentovat společnost.
- Využití stávajícího loga, jehož typické umístění na stránkách je v levém horním rohu.
- Z úvodní stránky musí být zřejmé, čím se společnost KS-Program zabývá.
- Přihlašování na stránky, vstup do košíku a uživatelského účtu pomocí grafického nebo textového prvku.
- Je třeba dbát na volbu optimálního řezu písma a na zobrazení textů v různých webových prohlížečích.
- Zachování obdobné barevné kombinace s převládající modrou a bílou barvou.
- Stránky musí mít flexibilní výšku, vedlejší menu v pravé části stránek.
- Celkový vzhled webu založen na jednoduchosti, přehlednosti a uživatelské přívětivosti.

3.3 Analýza

Smyslem analýzy je vysvětlit podstatu problému, vytvořit modely IS na úrovni problémové domény, zjednodušit složitější pohledy (co bude systém umět).

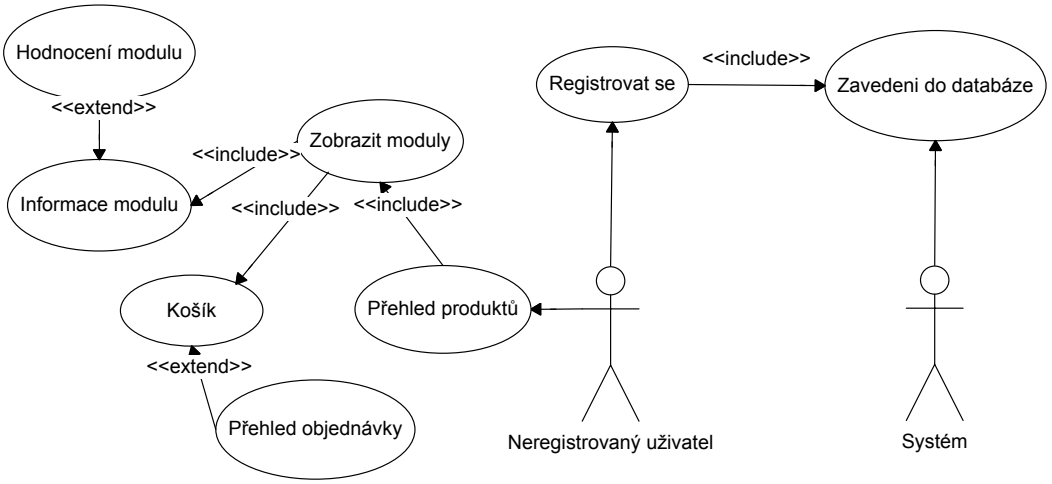
Analýze obvykle předchází specifikace požadavků, což je klíčový moment, kdy dochází ke styku uživatele s řešitelem, a ti musí najít společný jazyk.

Výstupem analýzy jsou modely, sloužící pro správné pochopení systému a pro komunikaci mezi uživatelem, analytikem a realizátorem. Obvykle používaným nástrojem analýzy IS jsou Use Case diagramy, které definují, co je vně vyvíjeného systému (aktéři) a co má být systémem prováděno (případ užití).

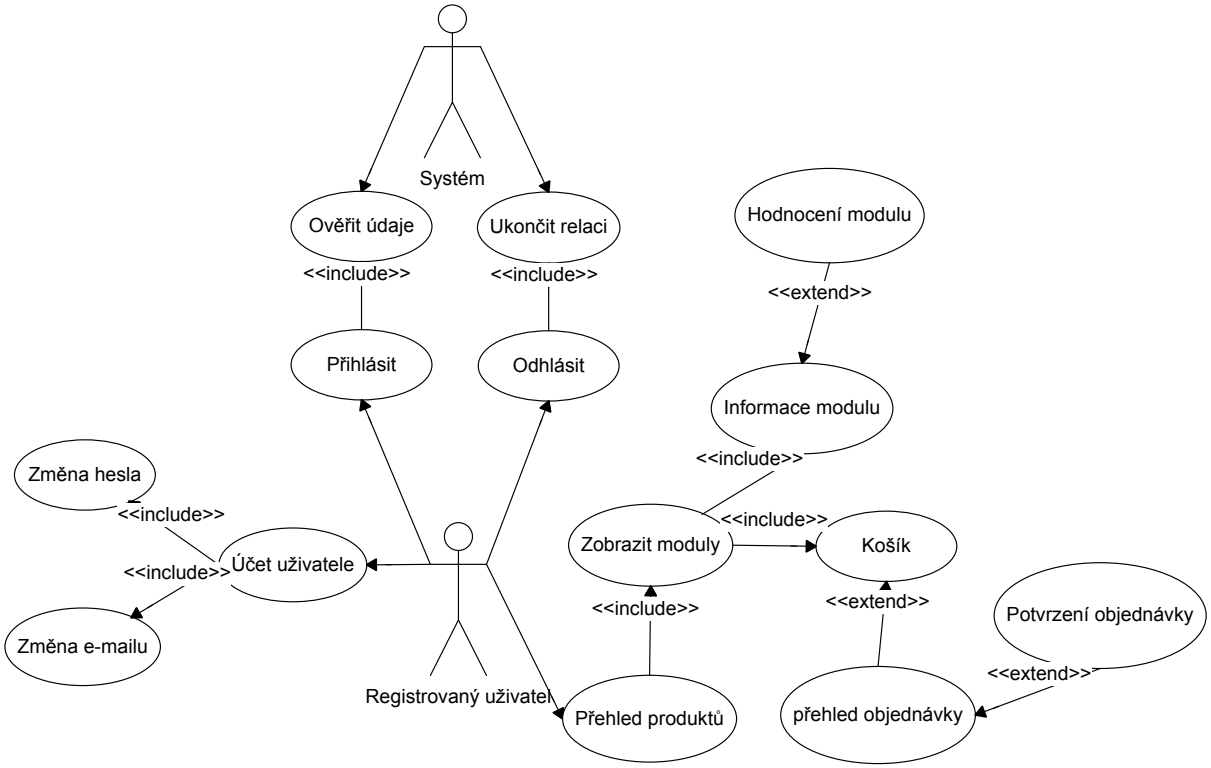
3.3.1 Use Case diagramy

Na základě zadaných požadavků a specifikace systému byly vytvořeny následující Use Case diagramy. První z Use Case diagramů zachycuje procesy pro neregistrovaného uživatele (obrázek 3.1). Následuje Use Case diagram registrovaného přihlášeného uživatele (obrázek 3.2). Poslední Use Case diagram ukazuje procesy uživatele přihlášeného jako administrátor (obrázek 3.3).

V Use Case diagramech jsou použity relace typu <<include>> a relace typu <<extend>>. První z relací se objevuje tam, kde existuje podobná nebo stejná část sekvence scénáře, opakující se ve více případech užití. Druhý typ relace přidává rozšiřující případ užití nové – doplňkové chování do základního případu užití.

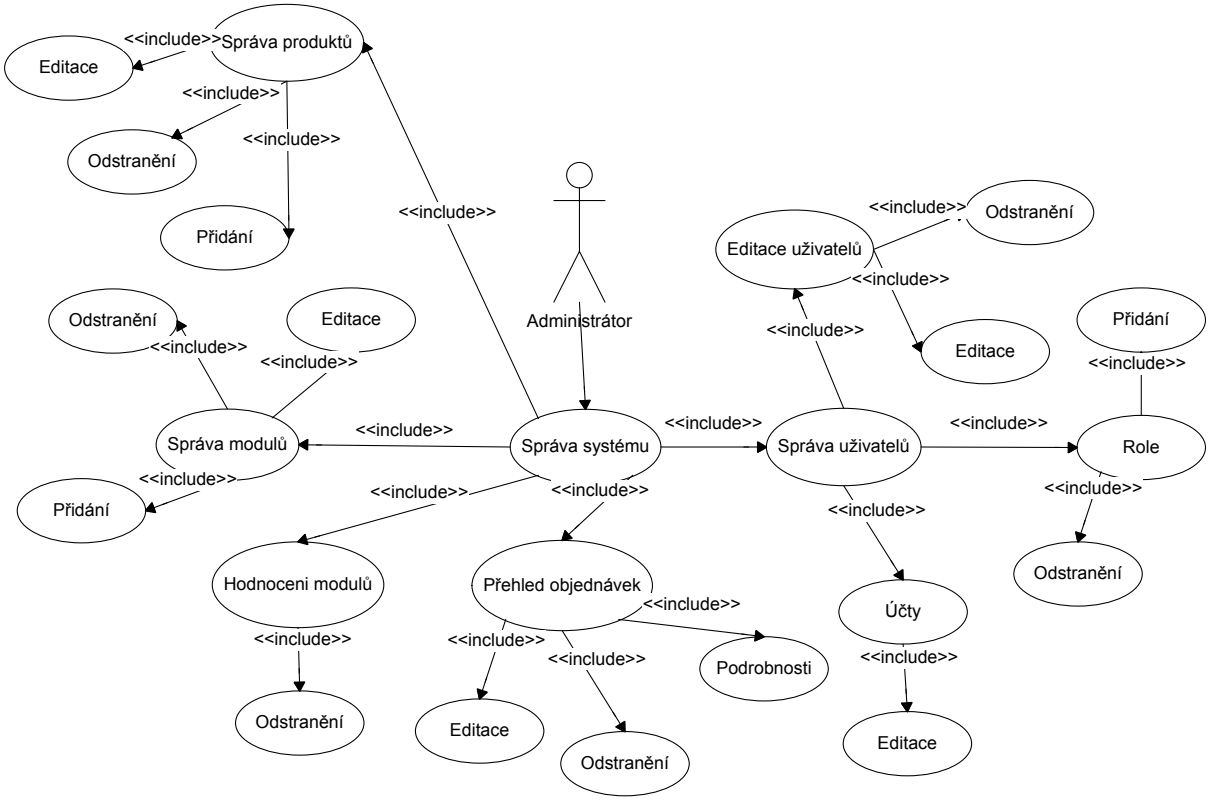


Obrázek 3.1 Use Case neregistrovaného uživatele



Obrázek 3.2 Use Case přihlášeného uživatele

Následující Use Case diagram zobrazuje procesy administrátora. Administrátor disponuje stejnými procesy jako registrovaný uživatel (viz. obrázek 3.3). Tyto procesy jsem již nezobrazoval. V Use Case diagramu níže uvádím tedy pouze rozšiřující procesy administrátora.



Obrázek 3.3 Use Case administrátora

3.3.2 Případy užití

Pro lepší představu předcházejících Use Case diagramů a pochopení fungování systému jsou uvedeny příklady popisu případu užití.

Název případu užití	**Registrace uživatele**
Identifikace případu užití	UC1
Aktéři	Neregistrovaný uživatel, Systém
Vstupní podmínky	Uživatel je neregistrovaný.
Kroky případu užití	1. Uživatel otevře stránku Vytvořit nový účet 2. Systém zobrazí formulář pro registraci uživatele. 3. Uživatel vyplní textová pole formuláře (uživatelské jméno, e-mail, heslo, potvrzení hesla) a odešle jej pomocí tlačítka „Vytvořit účet“.

	4. Systém ověří platnost zadaných údajů. Data jsou v pořádku. 5. Systém uloží údaje o uživateli do databáze a potvrdí úspěšné vložení informativním hlášením. 5a1. Systém zobrazí formulář pro registraci a upozorní na položky, které jsou špatně zadány. 5a2. Uživatel údaje opraví a znovu odešle formulář tlačítkem „Vytvořit účet“. 5a3. Systém uloží údaje o uživateli do databáze a potvrdí úspěšné vložení informativním hlášením.
Výstupní podmínky	Uživatel je zaregistrován.

Název případu užití	**Hodnocení modulu**
Identifikace případu užití	UC2
Aktéři	Neregistrovaný uživatel, Registrovaný uživatel, Systém.
Vstupní podmínky	-
Kroky případu užití	1. Uživatel otevře stránku vybraného modulu. 2. Systém zobrazí podrobné informace o modulu. ***Bod rozšíření: možnost přidání hodnocení konkrétnímu modulu.***
Výstupní podmínky	-

Název případu užití	**Nákupní košík**
Identifikace případu užití	UC3
Aktéři	Neregistrovaný uživatel, Registrovaný uživatel, Systém.
Vstupní podmínky	-

Kroky případu užití	1. Uživatel otevře stránku vybraného modulu. 2. Systém zobrazí podrobné informace o modulu. 3. Uživatel vybere akci „přidat do košíku“. 4. Systém zobrazí nákupní košík. ***Bod rozšíření: možnost pokračovat v nákupu a potvrzení objednávky.***
Výstupní podmínky	-

Název případu užití	**Nákup požadovaného modulu**
Identifikace případu užití	UC4
Aktéři	Registrovaný uživatel, Administrátor, Systém
Vstupní podmínky	Uživatel je registrovaný.
Kroky případu užití	1. Uživatel zvolí příslušný produkt. 2. Systém zobrazí seznam modulů příslušného produktu. 3. Uživatel vybere konkrétní modul nebo kompletní balík daného produktu. 4. Systém zobrazí podrobné informace daného modulu. 5. Uživatel vybere akci „Přidat do košíku“. 6. Systém vloží požadovaný modul do košíku 7. Systém zobrazí nákupní košík s příslušným modulem. 8. Uživatel vybere akci „Potvrdit“. 9. Systém zobrazí přehled objednávky a formulář pro vyplnění kontaktních údajů. 10. Uživatel vyplní textová pole a odešle formulář pomocí tlačítka „Potvrdit“. 11. Systém uloží data do databáze a zobrazí informativní hlášení o úspěšném odeslání objednávky.

	11a1. Systém zobrazí formulář pro vyplnění kontaktních údajů spolu se špatně zadanými položkami. 11a2. Uživatel údaje opraví a znovu odešle formulář tlačítkem „Potvrdit“. 11a3. Systém uloží data do databáze a zobrazí informativní hlášení o úspěšném odeslání objednávky.
Výstupní podmínky	Úspěšně odeslaná objednávka.

Název případu užití	**Úpravy v nákupním košíku**
Identifikace případu užití	UC5
Aktéři	Registrovaný uživatel, Administrátor, Systém
Vstupní podmínky	Uživatel je registrovaný.
Kroky případu užití	1. Uživatel vybere konkrétní modul. 2. Systém zobrazí podrobné informace o modulu. 3. Uživatel vybere akci „přidat do košíku“. 4. Systém vloží požadovaný modul do košíku 5. Systém zobrazí nákupní košík s příslušným modulem. 6. Uživatel změní požadované množství daného modulu a potvrdí změnu tlačítkem „Aktualizovat košík“. 7. Systém uloží změny a aktualizuje nákupní košík. 8. Uživatel zaškrtne políčko u příslušného modulu, v případě že požaduje odstranění modulu a potvrdí odstranění tlačítkem „Aktualizovat košík“. 9. Systém modul odstraní a aktualizuje nákupní košík.

Výstupní podmínky	Upravený nákupní košík podle potřeb zákazníka

3.4 Datová analýza

Hlavním úkolem datové analýzy je vytvoření modelu budoucí webové aplikace z pohledu datových struktur, se kterými bude aplikace pracovat.

Datová analýza zahrnuje následující kroky:

- lineární zápis,
- ER-diagram,
- popis atributů.

3.4.1 Lineární zápis

Cílem lineárního zápisu je popsat entity a vazby prostřednictvím klíčových atributů.

Slova před závorkou označují názvy jednotlivých tabulek v databázi.

Tučně zvýrazněná slova v závorce přestavují **primární klíč**. Primární klíč je pole nebo kombinace polí jednoznačně identifikující každý záznam v databázové tabulce. Žádné pole, které je součástí primárního klíče, nesmí obsahovat hodnotu NULL. Každá tabulka má definovaný právě jeden primární klíč.

Slova v závorce **podtržená** přestavují **cizí klíč**. Cizí klíč je v prostředí relačních databází integritní omezení, které u tabulky vytvoří spojení jednoho nebo více jejích sloupců se sloupcem nebo sloupci jiné („cizí“) tabulky. Pokud se hodnoty dotčených sloupců shodují, poté příslušný řádek cizí tabulky rozvíjí řádek zdrojové tabulky přes toto spojení.

Produtky (**ProduktID**, NazevProduktu)

Moduly (**ModulID**, ProduktID, CisloModulu, NazevModulu, Obrazek, Cena, Popis)

NakupniKosik (**ZaznamID**, KosikID, Monostvi, ModulID, Vytvoreni)

Recenze (**RecenzeID**, ModulID, ZakJmeno, ZakEmail, Hodnoceni, Komentar)

Objednavky (**ObjednavkaID**, ZakJmeno, DatumObjednavky, DatumOdeslani, Jmeno, Prijmeni, Adresa)

DetailyObjednavky (**Id**, ObjednavkaID, ModulID, Mnozstvi, Cena)

Odvetvi (**IDOdvetvi**, Nazev)

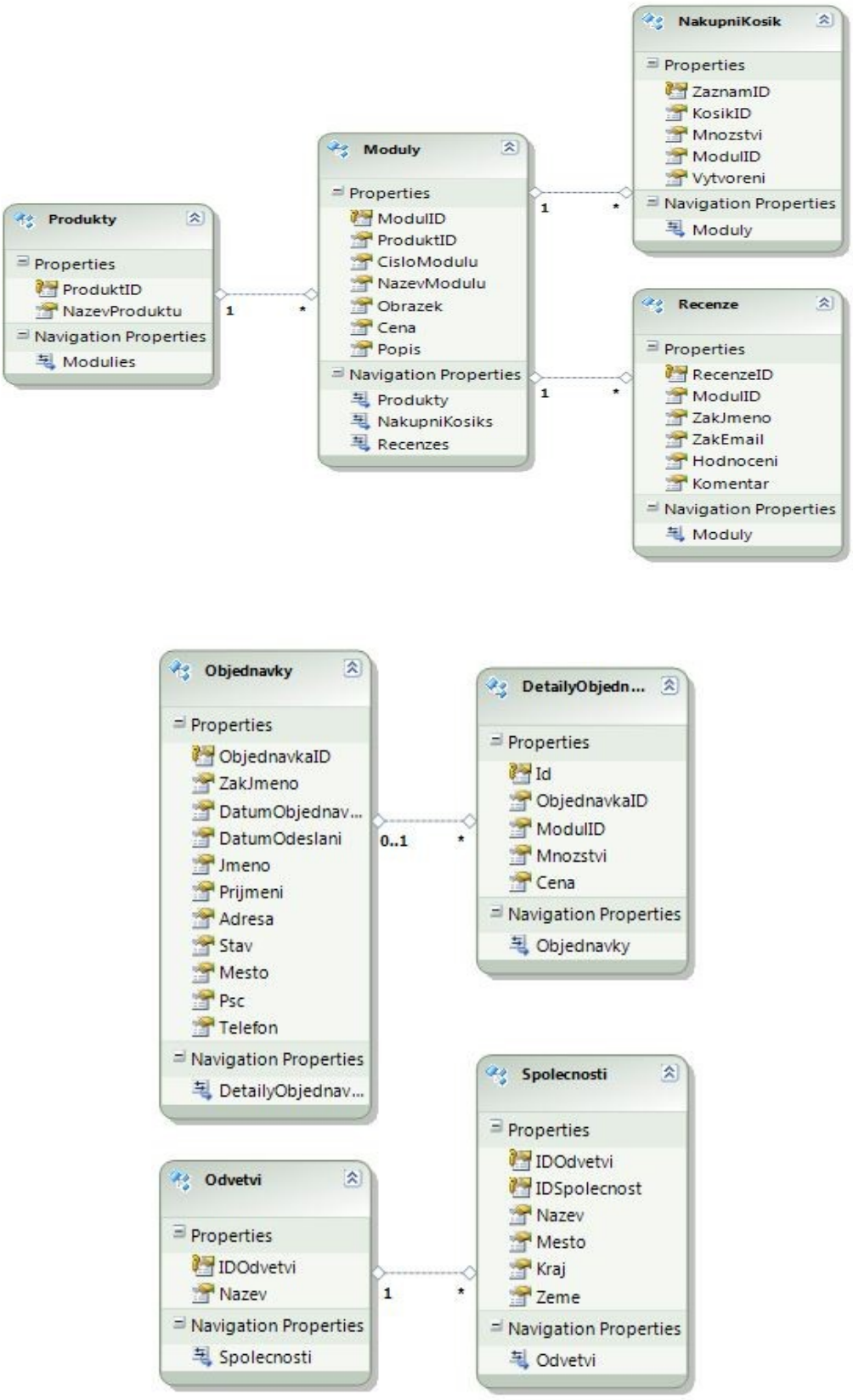
Spolecnosti (IDOdvetvi, **IDSpolecnost**, Nazev, Město, Kraj, Zeme)

Popis vztahů:
Nejpoužívanějším vztahem je vztah 1:N. Vyjadřuje vztah, kdy k jednomu záznamu v jedné tabulce existuje několik záznamů v tabulce druhé. Dalším vztahem je vztah M:N. Ten vyjadřuje vztah, kdy k několika záznamům (M) v jedné tabulce existuje několik záznamů (N) v tabulce druhé.

Obsahuje (Produkty, Moduly) 1:N
Vlozeni (Moduly, NakupniKosik) 1:N
Hodnoceni (Moduly, Recenze) 1:N
Podrobnosti (Objednavky, DetailyObjednavky) M:N
Zahrnuji (Odvetvi, Spolecnosti) 1:N

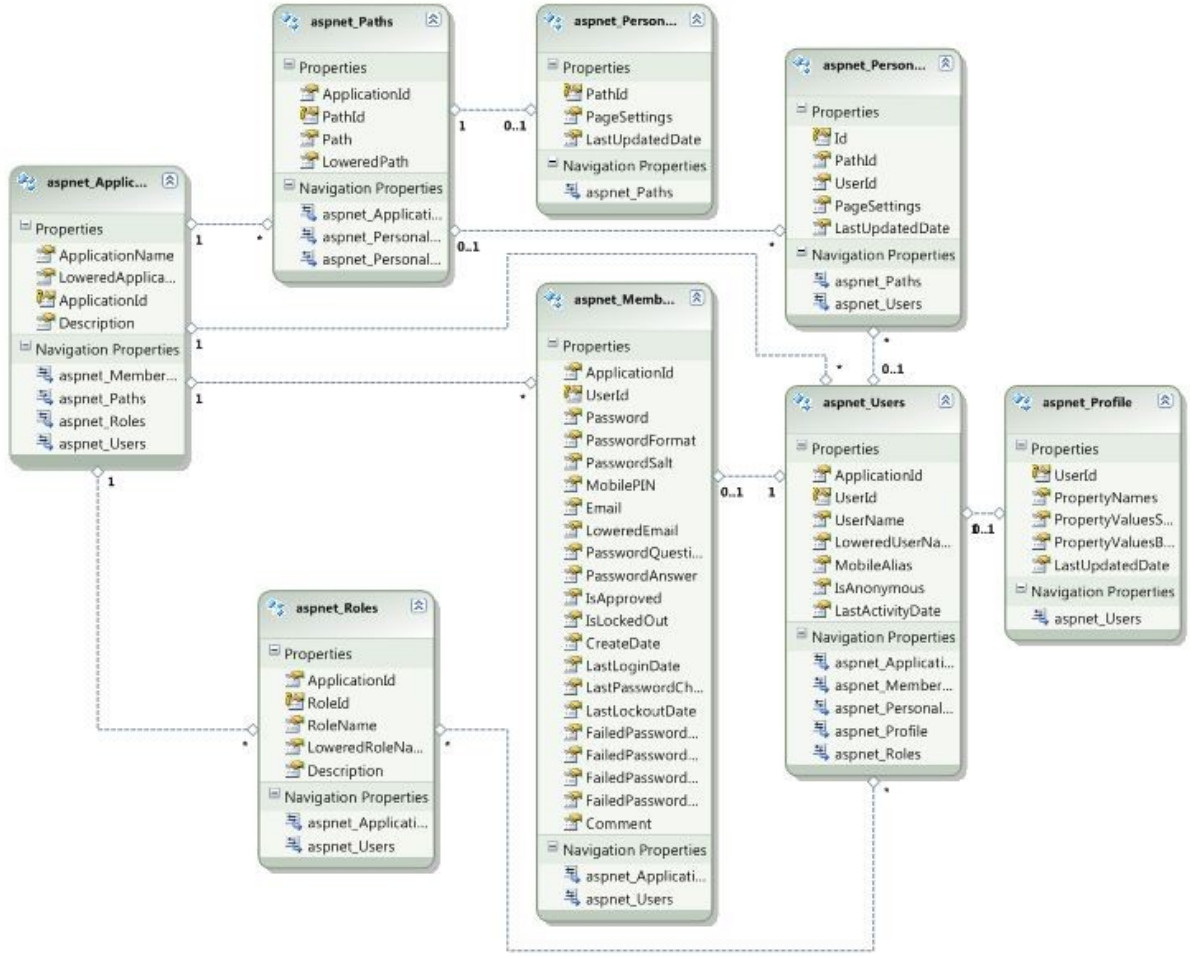
3.4.2 ER-diagram

ER- diagram pomáhá, na konceptuální úrovni abstrakce, popsat uživatelskou aplikaci za účelem specifikovat následně strukturu databáze. Na obrázku 3.4 je ER-diagram webové aplikace.



Obrázek 3.4 ER-diagram webové aplikace
zdroj: Microsoft Visual Web Developer 2010

Na obrázku 3.5 je zobrazen ER-diagram API členství. API členství je pracovní rámec, který poskytuje kompletní sadu funkcí pro správu uživatelů. Není tedy předmětem vývoje webové aplikace, ale jeho funkcionality jsou využívány.



Obrázek 3.5 ER-diagram API členství
zdroj: Microsoft Visual Web Developer 2010

3.4.3 Popis atributů

Produkty

Název	Typ	Délka	Klíč	Not Null	Popis
ProduktID	int		PK		Identifikační číslo produktu
NazevProduktu	nvarchar	50		Y	Název produktu

Tabulka 3.1 Produkty

Moduly

Název	Typ	Délka	Klíč	Not Null	Popis
ModulID	int		PK		Identifikační číslo modulu
ProduktID	int		FK		Identifikační číslo produktu
CisloModulu	nvarchar	10		Y	Popisující číslo modulu
NazevModulu	nvarchar	50		Y	Název modulu
Obrazek	nvarchar	50		Y	Název obrázku
Cena	money				Cena modulu
Popis	nvarchar	3000		Y	Dolpňující popis modulu

Tabulka 3.1 Moduly

NakupniKosik

Název	Typ	Délka	Klíč	Not Null	Popis
ZaznamID	Int		PK		Pořadové číslo záznamu v košíku
KosikID	nvarchar	50			Identifikační číslo košíku
Mnozstvi	Int				Nakoupené množství
ModulID	Int		FK		Identifikační číslo modulu
Vytvoreni	datetime				Datum vložení do košíku

Tabulka 3.2 Nákupní košík

Recenze

Název	Typ	Délka	Klíč	Not Null	Popis
RecenzeID	Int		PK		Identifikační číslo recenze
ModulID	Int		FK		Identifikační číslo modulu
ZakJmeno	nvarchar	50		Y	Uživatelské jméno
ZakEmail	nvarchar	50		Y	E-mail uživatele
Hodnoceni	Int				Hodnocení modulu
Komentar	nvarchar	3000		Y	Komentář

Tabulka 3.3 Recenze

Objednavky

Název	Typ	Délka	Klíč	Not Null	Popis
ObjednavkaID	int		PK		Identifikační číslo objednávky
ZakJmeno	nvarchar	50		Y	Uživatelské jméno
DatumObjednavky	datetime				Datum vytvoření objednávky
DatumOdeslani	datetime				Datum do kdy má být objednávka odeslána
Stav	nvarchar	20		Y	Udává stav objednávky
Jmeno	nvarchar	20		Y	Jméno zákazníka
Prijmeni	nvarchar	20		Y	Příjmení zákazníka
Adresa	nvarchar	30		Y	Adresa zákazníka
Mesto	nvarchar	30		Y	Město
Psc	nvarchar	5		Y	Poštovní směrovací číslo
Telefon	nvarchar	15		Y	Telefon zákazníka

Tabulka 3.4 Objednávky

DetailyObjednavky

Název	Typ	Délka	Klíč	Not Null	Popis
Id	int		PK		Identifikační číslo
ObjednavkaID	int		FK	Y	Identifikační číslo objednávky
ModulID	int			Y	Identifikační číslo modulu
Mnozstvi	int			Y	Objednané množství
Cena	money			Y	Cena za modul

Tabulka 3.5 Detaily objednávky

Odvetvi

Název	Typ	Délka	Klíč	Not Null	Popis
IDOdvetvi	int		PK		Identifikační číslo odvětví
Nazev	nvarchar	70		Y	Název odvětví

Tabulka 3.6 Odvětví

Spolecnosti

Název	Typ	Délka	Klíč	Not Null	Popis
IDOdvetvi	int		FK		Identifikační číslo odvětví
IDSpolecnosti	int		PK		Identifikační číslo společnosti
Nazev	nvarchar	80		Y	Název společnosti
Mesto	nvarchar	50		Y	Město
Kraj	nvarchar	10		Y	Kraj
Zeme	nvarchar	10		Y	Země

Tabulka 3.7 Společnosti

4. Návrh webové aplikace

V této části diplomové práce je specifikováno vývojové a aplikační prostředí, definovány základní prvky použité při vývoji webové aplikace, popsány základní charakteristiky jednotlivých webových formulářů a seznámení se s vzhledem webové aplikace.

4.1 Implementace

Základním předpokladem fungování webové aplikace je architektura klient-server. Server je zde chápán, jako počítač či skupina počítačů, na kterých běží celý systém a provádějí se zde databázové a funkční operace. Klientem je pak počítač, který se serverem komunikuje prostřednictvím internetu či intranetu a jediným jeho úkolem je komunikace s uživatelem pomocí klientského software (internetového prohlížeče).

Pro implementaci a vývoj celé aplikace byla zvolena platforma Microsoft .NET 4.0, konkrétně prostředí pro tvorbu webových aplikací ASP.NET 4.0. Vývojovým prostředím se stalo Microsoft Visual Studio 2010 Express, což je komplexní vývojové prostředí aplikace obsahující nejen výkonný editor zdrojových kódů, ale také nabízí rozšířené funkce, jako je ladění, krokování atd. Programovacím jazykem byl zvolen C#.

4.2 Vývoj založený na projektu

Visual Studio nabízí dva způsoby, jak vytvořit webovou aplikaci. Vývoj založený na projektu a vývoj bez projektu. Pro řešení diplomové práce byl vybrán první způsob, který v sobě nese řadu výhod. Jednou z nich je to, že při spuštění webového projektu, Visual Studio zkompiluje veškerý kód do jediné assembly ještě předtím, než se spustí webový prohlížeč. To má za následek snížení zátěže vznikající při prvním požadavku jako je to u vývoje založeného bez projektu. Dalšími výhodami jsou flexibilnější správa souborů, striktnější vývoj oproti vývoji bez projektu, využití modelu kódu v pozadí a deklaraci webových prvků. [4]

4.3 Adresáře

Výchozí projekt obsahuje následující adresáře:

- Account – implementuje základní uživatelské rozhraní pro membership subsystém ASP.NETu.
- App_Data – obsahuje vytvořené databáze.
- Scripts – slouží jako úložiště pro JavaScript soubory na straně klienta.
- Styles – obsahuje soubory zabezpečující vzhled webových stránek (Kaskádové styly, fonty, obrázky…).

Adresáře, které byly vytvořeny v průběhu vývoje aplikace:

- Administrator - obsahuje webové formuláře, ke kterým má přístup pouze správce aplikace.
- Clasess - implementuje obchodní logiku webové aplikace.
- Controls - zde jsou uloženy uživatelské ovládací prvky.
- Data_access - obsahuje vrstvu datového přístupu k databázi vytvořené na základě .NET Entity Framework.
- Katalog - slouží jako úložiště pro obrázky jednotlivých produktů.

4.4 Vrstva datového přístupu

Webová aplikace obsahuje dvě databáze. Pro informace o uživatelích je použita klasická databáze API členství, která je součástí ASP.NET. Pro nákupní košík, katalog produktů a modulů je vytvořena samostatná databáze. Její začlenění do

webové aplikace je zabezpečeno použitím .NET Entity Framework. Do složky „Data_access“ je přidána nová položka prostřednictvím “ADO.NET Entity Data Model”.

K vygenerování databáze pro správu uživatelů jsem využil Visual Studio. Prostřednictvím webové konfigurační utility ASP.NET je vyvoláno prostředí WAT (Web Site Administration Tool), kde je možno přidávat uživatele. Jakmile vytvoříme uživatele, vytvoří se automaticky požadované podkladové úložiště s názvem ASPNETDB uložené ve složce App_Data. Tato databáze implementuje kompletní schéma, které je potřebné pro ukládání a správu informací o uživatelích, informací o rolích, přiřazení uživatelů k rolím a další.

4.5 Formulářová autentizace a role

Kořenový adresář webové aplikace často uděluje přístup anonymním uživatelům. Protože kořenový adresář obsahuje webové formuláře, u kterých je nutné odepřít přístup anonymním uživatelům, je třeba nakonfigurovat soubor web.config. Tímto zápisem odepřeme přístup anonymním uživatelům k následujícím webovým formulářům:

```
<location path="Potvrzeni.aspx">
   <system.web>
     <authorization>
       <deny users="?" />
     </authorization>
   </system.web>
 </location>
 <location path="Recenze.aspx">
   <system.web>
     <authorization>
       <deny users="?" />
     </authorization>
   </system.web>
 </location>
```

Pokud se nepřihlášený uživatel bude chtít dostat k těmto formulářům, runtime ASP.NET automaticky takového uživatele přesměruje na veřejně dostupnou přihlašovací stránku.

Oddělení administrátorské sekce od zbytku aplikace je zabezpečeno prostřednictvím role. Všechny webové formuláře určené pro administrátora se nacházejí v adresáři „Administrator“. Přístup k obsahu tohoto adresáře je nakonfigurován pomocí

přístupových pravidel v rámci WAT (Web Site Administration Tool). Pro zobrazení odlišných ovládacích prvků pro anonymní uživatele a pro přihlášené uživatele je v rámci aplikace použit ovládací prvek LoginView. Tento ovládací prvek podporuje šablonu, která umožňuje vytvářet rozdílné pohledy v závislosti na rolích, ke kterým uživatel patří. Za tímto účelem je použita šablona RoleGroups s ovládacími prvky <asp:RoleGroup>. Uvnitř každého ovládacího prvku je specifikována prostřednictvím atributu Roles role, pro které bude zobrazena šablona <ContentTemplate>. [4] Použití ovládacího prvku LoginView je vidět v následujícím kódu:

```
<asp:LoginView ID="LoginView1" runat="server">
        <RoleGroups>
           <asp:RoleGroup Roles="Spravce">
              <ContentTemplate>
                 <li class="last2"><a href="Administrator/Prehled.aspx">Administrace</a></li>
              </ContentTemplate>
           </asp:RoleGroup>
        </RoleGroups>
</asp:LoginView>
```

4.6 Master page

Protože bylo třeba zajistit, aby na všech stránkách bylo zabezpečeno stejné rozvržení (layout), bylo pro tyto účely použito vzorů stránek (master pages). Jde vlastně o šablony pro webovou stránku, které definují fixní obsah a deklarují tu část webové stránky, kam se vkládá vlastní obsah.

V diplomové práci je využit hlavní vzor stránky (master page) a její vnořený vzor (nested master page). Je to z toho důvodu, že stránky webové aplikace se liší ve struktuře a zobrazovaném obsahu.

Ve vzoru stránky a vnořeném vzoru stránky (master page, nested master pages) je umístěno několik zástupců obsahu (content placeholders), což jsou předdefinované části, které se mohou modifikovat. Každá obsahová stránka (content page) se odkazuje na jediný vzor stránek a přebírá z tohoto vzoru nejenom samotné rozvržení (layout), ale také obsah. Každá obsahová stránka může přidávat do libovolných zástupců obsahu svůj vlastní obsah, specifický pro danou stránku. [5]

4.7 Webové formuláře

Webové formuláře jsou životně důležitou částí aplikace ASP.NET. Poskytují skutečný výstup webové aplikace. Jde o stránky, které požadují klienti a dívají se na ně prostřednictvím některého z internetových prohlížečů.

V této části jsou stručně popsány jednotlivé formuláře webové aplikace. K některým formulářům je omezen přístup prostřednictvím přístupových rolí. Uživatel v roli správce může přistupovat ke všem formulářům. Uživatelé přihlášení a neregistrovaní mohou přistupovat ke všem formulářům mimo formuláře určené pro uživatele v roli správce. Zde je jejich výpis a popis:

Webové formuláře ve složce Account:

- **Login.aspx** – zobrazuje přihlašovací formulář s textovými poli pro uživatelské jméno a heslo. Dále pak odkazy na stránky registrace a zapomenutého hesla.
- **Register.aspx** – zobrazuje formulář pro registraci uživatele. Obsahuje textová pole uživatelské jméno, email, heslo a potvrzení hesla.
- **Ucet.aspx** – na tento formulář se dostane pouze registrovaný uživatel, kterému zobrazí informace o jeho účtu s odkazy na změnu hesla a emailu. (Pouze administrátor a přihlášený uživatel).
- **ZapomenuteHeslo.aspx** – webový formulář, na kterém je uživatel dotázán na uživatelské jméno, na jehož základě obdrží do svého emailu nově vygenerované heslo.
- **ChangeEmail.aspx** – webový formulář, ve kterém si uživatel může změnit email. (Pouze administrátor a přihlášený uživatel).
- **ChangePassword.aspx** – zobrazí formulář, kde na základě vložení starého hesla a vložení nového a potvrzení nového hesla dojde ke změně tohoto hesla. (Pouze administrátor a přihlášený uživatel).
- **ChangePasswordSuccess.aspx** – zobrazí informační hlášení o úspěšné změně hesla.

Webové formuláře ve složce Administrator:

- **Prehled.aspx** – úvodní formulář administrátorské sekce.

- **Produkty.aspx** – webový formulář zobrazující v tabulce veškeré produkty (ProduktID, název produktu) s možností jejich úpravy, odstranění nebo přidání nového produktu.
- **PridatProdukt.aspx** – webový formulář sloužící k zavedení nového produktu do databáze.
- **Moduly.aspx** - webový formulář zobrazující v tabulce veškeré moduly (ID produktu, číslo modulu, název, obrázek, cena, popis) s možností jejich úpravy, odstranění nebo přidání nového produktu.
- **PridatModul.aspx** - webový formulář obsahující náležitá textová pole sloužící k zavedení nového modulu do databáze.
- **Recenze.aspx** – zobrazuje hodnocení a komentáře konkrétních modulů s možností jejich smazání.
- **Objednavky.aspx** – webový formulář zobrazující veškeré objednávky s možností jejich úpravy, odstranění a zobrazení podrobností konkrétní objednávky.
- **PodrobnostiObjednavky.aspx** – tento webový formulář zobrazuje souhrnné informace o objednávce.
- **SpravaUzivatelu.aspx** – webový formulář zobrazující úvodní obrazovku pro správu uživatelů s odkazy do jednotlivých sekcí (účty, editace, role uživatelů).
- **Uzivatele.aspx** – webový formulář zobrazující přehled všech uživatelů a jejich stavy účtů (schválený, zamčený účet, přihlášený uživatel). Umožňuje filtrovat uživatele podle počátečního písmena.
- **PodrobnostiUzivatele.aspx** – tento webový formulář umožňuje odsouhlasit účet uživateli a umožnit tak jeho přístup nebo odemknout účet.
- **EditovaUzivatele.aspx** – zobrazuje přehled uživatelů s možností jejich úpravy a odstraňování. Umožňuje filtrovat uživatele podle počátečního písmena.
- **UzivateleARole.aspx** – webový formulář sloužící k zobrazení uživatelů v příslušné roli, jejich odebrání z role nebo přidání do konkrétní role.

Ostatní webové formuláře v projektu:

- **Default.aspx** – webový formulář zobrazující úvodní informace (aktuality, informace, produkty) jde o domovskou stránku.
- **Aktuality.aspx** – tento formulář zobrazuje aktuality z oblasti zájmu společnosti prostřednictvím Ajax prvku AccordionPane.
- **JakNakupovat.aspx** – informuje o způsobu nakupování v e-shopu.
- **ObchodniPodminky.aspx** – informuje o obchodních podmínkách v rámci elektronického obchodu.
- **Uhrada.aspx** – informuje o možných způsobech úhrady v e-shopu.
- **Produkty.aspx** – webový formulář zobrazující přehled a popis jednotlivých produktů pomocí Ajax prvku AccordionPane.
- **Sluzby.aspx** – webový formulář, který zobrazuje a informuje o nabízených službách společnosti.
- **O_nas.aspx** – webový formulář poskytující stručnou historii o firmě.
- **Reference.aspx** – webový formulář, který na základě výběru odvětví z rozbalovacího seznamu zobrazí příslušné reference.
- **Kontakt.aspx** – webový formulář zobrazující kontaktní údaje společnosti a formulář pro odesílání případných dotazů.
- **SeznamModulu.aspx** – zobrazuje moduly příslušného softwarového produktu. Každý modul je zobrazen jako kartička s příslušným názvem, který také slouží jako odkaz pro získání více informací o daném modulu, dále označením, cenou a rychlým odkazem umožňující přidat konkrétní modul do košíku.
- **PopisModulu.aspx** – webový formulář zobrazující podrobné informace o příslušném modulu, jeho označení, cenu, odkaz na přidání do košíku a odkaz na hodnocení daného modulu případně tabulku s hodnoceními a komentáři.

- **Recenze.aspx** – umožňuje hodnotit konkrétní modul. Obsahuje textové pole pro uživatelské jméno, email, hodnocení v podobě hvězdiček od jedné do pěti a textové pole pro přidání komentáře.
- **PridatDoKosiku.aspx** – webový formulář, který zobrazuje aktuální košík s právě přidanými položkami.
- **MujNakupniKosik.aspx** – webový formulář, který zobrazuje nákupní košík s vybraným zbožím. V košíku lze měnit počet kusů daného zboží nebo jej z košíku odstranit, vše je třeba provádět pomocí tlačítka „Aktualizovat košík“. Dále je tu tlačítko „Potvrdit“ sloužící k potvrzení nákupu.
- **Potvrzeni.aspx** – zobrazí vybrané zboží, formulář pro zadání poštovní adresy a tlačítko na konečné potvrzení a závazné odeslání objednávky.

4.8 Obchodní logika

Hlavním cílem je nabídnout nákupní možnosti vždy, když někdo navštíví webovou aplikaci. Návštěvníci budou moci procházet a přidávat položky do nákupního košíku, i když nejsou registrováni nebo přihlášeni. Pokud ale bude uživatel chtít potvrdit objednávku a nebude přihlášený, bude přesměrován na stránku pro přihlášení, případně stránku pro vytvoření uživatelského účtu.

Tuto logiku tvoří třída „Class“ pojmenovaná jako MujNakupniKosik.cs. Třída je rozšířena o implementaci stránky MujNakupniKosik.aspx prostřednictvím .NET konstruktoru „Partial class“. Implementace je provedena následovně:

```
namespace ESLwebform
{
        public partial class MujNakupniKosik
        {
        }
}
```

Třída MujNakupniKosik.cs obsahuje několik metod. Jednou z nich je například metoda „AddItem“. Jedná se o přidání položky do košíku. Tato metoda je volána vždy, když uživatel klikne na tlačítko „přidat do košíku“ v seznamu modulů či na stránce „popis modulu“.

```
public void AddItem(string kosikID, int modulID, int mnozstvi)
    {
      using (ESLEntities db = new ESLEntities())
```

```
        {
            try
            {
              var polozka = (from c in db.NakupniKosiks where c.KosikID == kosikID && c.ModulID == modulID
            select c)
              .FirstOrDefault();
              if (polozka == null)
                {
                    NakupniKosik pridat = new NakupniKosik();
                    pridat.KosikID = kosikID;
                    pridat.ModulID = modulID;
                    pridat.Mnozstvi = mnozstvi;
                    pridat.Vytvoreni = DateTime.Now;
                    db.NakupniKosiks.AddObject(pridat);
                }
                else
                {
                    polozka.Mnozstvi += mnozstvi;
                }

                db.SaveChanges();
            }
            catch (Exception exp)
            {
                throw new Exception("ERROR: Nelze přidat položku do košíku - " + exp.Message.ToString(), exp);

            }
        }
 }
```

K zjištění, zda je položka již v košíku, je použit LINQ dotaz nad entitou databáze. Pokud položka existuje, zvýší se množství objednávané položky. Pokud ne, vytvoří se nová položka. Aby bylo možné volat tuto metodu je implementována stránka PridatDoKosiku.aspx, která nejenom volá tuto metodu, ale také zobrazuje aktuální košík s právě přidanými položkami. Kód stránky je zobrazen níže.

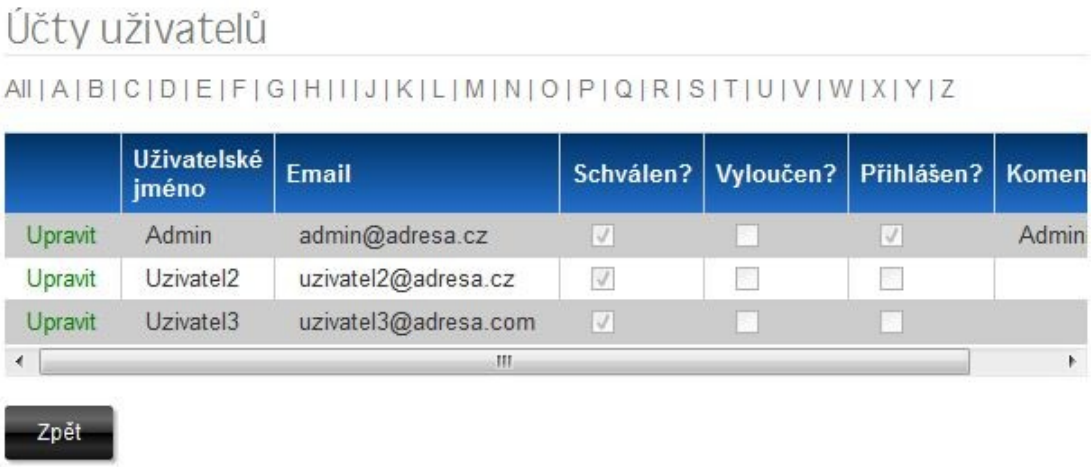
```
namespace ESLwebform
{
    public partial class PridatDoKosiku : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (Request.Params["ModulID"] != null)
            {
                MujNakupniKosik kosikUzivatele = new MujNakupniKosik();
                String kosId = kosikUzivatele.GetNakupniKosikId();
                kosikUzivatele.AddItem(kosId, Int32.Parse(Request.Params["ModulID"]), 1);
            }

            Response.Redirect("MujNakupniKosik.aspx");
        }
    }
}
```

Na základě parametru „ModulID“ a volání metody GetNakupniKosikId() obsažené ve třídě MujNakupniKosik.cs je zobrazen aktuální košík uživatele s nakoupenými položkami.

4.9 Administrátorská sekce

Do této části webové aplikace má výhradní přístup uživatel, který vystupuje v roli pojmenované jako „Spravce“. Tento přístup je zabezpečen pomocí ovládacího prvku LoginView a jeho šablony RoleGroups. Uživatel v této roli může provádět činnosti od přidávání produktů, modulů, sledování objednávek až po správu uživatelů. Popis veškerých webových formulářů spadající pod pravomoc „Spravce“ jsou popsány v předcházející kapitole.

Společným prvkem všech webových formulářů v této sekci je ovládací prvek GridView. Tento prvek je zvolen díky jeho potřebným zobrazovacím možnostem a široké škále zabudovaných schopností. Umožňuje flexibilně zobrazovat data v mřížce, stránkovat, řadit, editovat a rozšiřovat prostřednictvím šablon. Na následujícím obrázku 4.1 je podoba tohoto prvku ve webovém formuláři.



Obrázek 4.1 Účty uživatelů
zdroj: Martin Hatlapatka – vytvořeno v rámci diplomové práce

Syntaxe tohoto prvku vypadá následovně:

```
<asp:GridView ID="UserAccounts" runat="server" AutoGenerateColumns="False" BorderColor="#CCCCCC"
            ViewStateMode="Disabled">
        <Columns>
          <asp:HyperLinkField DataNavigateUrlFields="UserName"
                      DataNavigateUrlFormatString="PodrobnostiUzivatele.aspx?user={0}"
                      Text="Upravit" ControlStyle-ForeColor="Green" />
          <asp:BoundField DataField="UserName" HeaderText="Uživatelské jméno"/>
          <asp:BoundField DataField="Email" HeaderText="Email" />
          <asp:CheckBoxField DataField="IsApproved" HeaderText="Schválen?"/>
          <asp:CheckBoxField DataField="IsLockedOut" HeaderText="Vyloučen?" />
          <asp:CheckBoxField DataField="IsOnline" HeaderText="Přihlášen?"/>
          <asp:BoundField DataField="Comment" HeaderText="Komentář"/>
```

```
    </Columns>
    <HeaderStyle CssClass="adminTableHeader" Font-Bold="True" ForeColor="White" />
    <AlternatingRowStyle BackColor="White" />
    <RowStyle BackColor="#cccccc" ForeColor="#333333" />
</asp:GridView>
```

Jedná se o webový formulář „Uzivatele.aspx“. Pomocí zmíněného ovládacího prvku GridView jsou zde zobrazeni všichni uživatelé. Uživatele je možno filtrovat kliknutím na písmeno nad tabulkou, představující první písmeno v uživatelském jméně. Dále má správce možnost měnit statusy u konkrétních uživatelů. Status „schválen“ informuje o tom, zda má nově registrovaný uživatel možnost se přihlásit ke svému účtu. Ve výchozím stavu je tato hodnota povolena a není třeba se dožadovat správce o ověření účtu a jeho povolení. Status „vyloučen“ se aktivuje tehdy, pokud se uživatel snaží o nepovolený přístup (zadá-li několikrát po sobě špatné heslo). Status „přihlášen“ zobrazuje, který z uživatelů je právě přihlášen.

4.10 Vzhled

Webovou aplikaci je možné provozovat na kterémkoliv operačním systému s internetovým prohlížečem. Jelikož je webová aplikace vyvíjena ve vývojovém prostředí Visual Studia, které disponuje nástroji pro zajištění a udržení kompatibility a standardů webových stránek (XHTML, HTML, CSS, …), je proto možné zobrazit stejné rozvržení stránek v těchto prohlížečích:

- Mozzila Firefox 3.6 a vyšší,
- Google Chrome 10 a vyšší,
- Internet Explorer 8 a vyšší.

Na obrázcích v příloze (obrázek A.1, obrázek A.2) je zobrazen vzhled webu. První obrázek ukazuje vzhled úvodní stránky. Ta je rozdělena do několika částí. Část pro zobrazení uživatelského menu; část pro zobrazení loga společnosti a hlavního menu („záhlaví“); dále část, která slouží pro prezentaci produktů společnosti; hlavní část pro zobrazení měnícího se obsahu („obsah“) a poslední část zobrazující další informace o firmě („zápatí“).

Na druhém obrázku (obrázek A.2) je vidět vzhled stránky využívající vnořenou master page. Ta přidává tzv. „drobečkovou navigaci“ a do pravé části sloupec s dalšími informacemi. Základní strukturu přebírá z hlavní master page.

4.11 Popis aplikace

Webová aplikace slouží jako web prezentující společnost KS-program, spol. s.r.o. a současně jako elektronický obchod. Jednotlivé stránky webové aplikace se skládají z několika částí. Ze záhlaví, zápatí a samotného obsahu. Úvodní stránka také obsahuje kontejner, ve kterém probíhá prezentace produktů. Ta je zajištěna prostřednictvím JavaScriptu. V rámci obsahu úvodní stránky jsou k dispozici základní informace o společnosti, informace o elektronickém obchodu, aktuality a kalendář událostí. Do ostatních stránek je přidán pravý sloupec a „drobečková“ navigace. Pravý sloupec obsahuje seznam jednotlivých produktů. Dále pro zvýšení prodeje zobrazuje 5 nejprodávanějších produktů, což je zabezpečeno díky uživatelskému ovládacímu prvku. A v neposlední řadě zobrazuje přehled novinek. Obsah stránek se mění v závislosti na právě využívajícím webovém formuláři. V rámci webu je možné se pohybovat prostřednictvím hlavního menu, které je umístěno v záhlaví každé stránky nebo prostřednictvím „drobečkové“ navigace. Menu pro přihlášení, zobrazení účtu a košíku je umístěno v pravé horní části záhlaví. Přihlášení uživatelé mají možnost nahlížet do nákupního košíku, zobrazovat si informace o svém účtu. V rámci svého účtu mohou měnit přihlašovací heslo nebo e-mail. Uživatel v roli správce má k dispozici administrátorskou sekci, ve které může spravovat webovou aplikaci.

5. Zhodnocení navrhovaného řešení

Podle mého názoru a také podle vyjádření zadavatele diplomové práce, webová aplikace svým rozsahem splňuje požadovanou funkčnost. Seznamuje uživatele se společností KS Program, spol. s.r.o. a s její činností. Přehledně nabízí produkty společnosti KS Program. Umožňuje uživateli sestavit objednávku zvolených produktů a seznámí jej se službami společnosti, které jsou spojeny jak s nasazením produktů u zákazníka, tak s následnou péčí o zákazníka.

Administrátorovi umožňuje webová aplikace sestavovat nabídku produktů a jejich modulů, sledovat objednávky a spravovat registrované uživatele.

Po technické stránce využívá aplikace moderní prvky ASP.NET technologie, které dodávají aplikaci dynamický a živý vzhled. Kladně je také hodnoceno použití technologie ADO.NET Entities, která nabízí komfortní práci s virtuální objektovou

databází.
Po grafické stránce je aplikace hodnocena kladně. Potěšil zejména design, který je barevně živý, ovšem ne přeplácaný. Barevné rozvržení a obrázky příjemně dokreslují web a přitom nezastiňují důležitý informační obsah webu, který zůstal ústředním bodem.

V aplikaci se podařilo dosáhnout všech požadavků stanovených na začátku vývoje a díky tomu je aplikace hodnocena jako povedená. V rámci budoucího rozvoje by bylo zajímavé rozšířit možnosti aplikace o jiné jazykové mutace, především o anglickou.

6. Závěr

Předmětem diplomové práce bylo vytvoření webové aplikace založené na technologii .NET pro firmu KS-program spol. s.r.o. Tato společnost se zabývá vývojem personálních a mzdových informačních systémů. Výsledkem je tedy aplikace prezentující zadavatelskou společnost, její produkty prostřednictvím jednoduchého elektronického obchodu a služby. Aplikaci je také možno spravovat v administrátorské sekci.

K tvorbě aplikace byla zvolena technologie ASP.NET 4.0, což je technologie od společnosti Microsoft pro vytváření webových aplikací běžících na straně serveru. Tato technologie byla vybrána ze dvou důvodů. Prvním důvodem je skutečnost, že firma tuto technologie již nějakou dobu využívá. Druhým důvodem je určitá znalost této technologie, kterou jsem získal během studia na VŠB. Diplomová práce je rozdělena do několika částí. První část (Základní teorie a vymezení pojmů technologie .NET) shrnuje základní vlastnosti jednotlivých technologií použitých při tvorbě webové aplikace. Mezi něž patří již zmíněná technologie ASP.NET dále ADO.NET, ASP.NET AJAX, CSS, UML. Cílem další části (Analýza a popis stávajícího stavu) je stanovení funkčních požadavků a analyzování systému k jeho lepšímu pochopení (co daný systém bude umět). Výsledkem pak jsou jednotlivé Use Case diagramy a případy užití. Součástí je i datová analýza, jejímž úkolem je vytvoření modelu budoucího informačního systému z pohledu datových struktur. V následující části (Návrh webové aplikace) popisuji jednotlivé kroky při tvorbě aplikace. Od vytvoření projektu až po popis vzhledu budoucí webové aplikace.

Poslední část (Zhodnocení navrhovaného řešení) je věnována zhodnocením navrhovaného řešení a nástinem případných zlepšení.

Výstupem diplomové práce je tedy firemní web, na který můžeme nahlížet ze dvou pohledů. Jako web prezentující firmu nebo web poskytující elektronický obchod se svou administrací. Na webu se mohou pohybovat tři typy uživatelů. Neregistrovaný uživatel, přihlášený uživatel a uživatel v roli správce. Přihlášení uživatelé mají vůči neregistrovaným uživatelům možnost dokončit požadovanou objednávku a hodnotit jednotlivé produkty nebo jejich moduly. Neregistrovaný uživatel je při pokusu o potvrzení objednávky přesměrován na stránku pro přihlášení respektive stránku pro vytvoření účtu. Uživateli přihlášenému jako správce je zpřístupněna administrátorská sekce. V této sekci má správce možnost zobrazit si přehledy všech produktů, modulů, objednávek nebo přehled uživatelů. S těmito přehledy může správce dále pracovat. Například editovat, mazat, přidávat, filtrovat uživatele, přidávat uživatele do rolí nebo jim zablokovat účet atd.

V průběhu vývoje webové aplikace jsem si rozšířil své znalosti o další techniky a nástroje, které Visual Studio nabízí. Seznámil jsem se se zázemím zadavatelské firmy a získal jsem tak nové zkušenosti, které mohu v praxi dále využít.

Seznam použité literatury

1. **AGARWAL VRAT, V. a HUDDLESTON, J.** *Databáze v C# 2008: Průvodce programátora.* 1. vyd. Brno : Computer Press, a.s., 2009. str. 424. ISBN 978-80-251-2309-6.
2. **KANISOVA, H. a MÜLLER, M.** *UML srozumitelně.* 1. vyd. Brno : Computer Press, 2004. str. 158. ISBN 80-251-0231-9.
3. **KOSEK, J.** XHTML - nová budoucnost pro jazyk HTML. *Vše o WWW.* [Online] [Citace: 12. 3. 2011.] Dostupný z WWW: <http://www.kosek.cz/clanky/xml/xml-xhtml.html>.
4. **McDONALD, M. a SZPUSZTA, M.** *Pro ASP.NET 3.5 in C# 2008: includes Silverlight 2.* 3rd. ed. Berkeley : Apress, 2009. str. 1478. ISBN 978-1-4302-1567-7.
5. **PÍSEK, S.** *ASP.NET začínáme programovat: podrobný průvodce začínajícího uživatele.* 1 vyd. Praha : Grada Publishing, 2003. str. 228. ISBN 80-247-0526-5.
6. **SCHMULLER, J.** *Knihovna programátora: myslíme v jazyku UML.* 1 vyd. Praha : Grada Publishing, 2001. str. 359. ISBN 80-247-0029-8.
7. **ŽDÁREK, R.** Úvod do JavaScriptu. *Tvorba webu, Tvorba WWW stránek.* [Online] [Citace: 12. 3. 2011.] Dostuný z WWW: <http://www.jakdelatweby.cz/javascript/uvod-do-javascriptu.php>.
8. HTML5 & CSS3 Support. *FindMeByIP.* [Online] 10. 7. 2010. [Citace: 10. 3. 2011.] Available from WWW: <http://www.findmebyip.com/litmus/#target-selector>.
9. *KS-PROGRAM, spol. s.r.o.* [Online] [Citace: 15. 3. 2011.] Dostupný z WWW: <http://www.ksprogram.cz/spolecnost>.
10. Web application. *PC Magazine.* [Online] [Citace: 10. 3. 2011.] Available from WWW: <http://www.pcmag.com/encyclopedia_term/0,2542,t=Web+application&i=54272,00.asp>.
11. Webová aplikace. *Wikipedie Otevřená encyklopedie.* [Online] 19. 1. 2011. [Citace: 10. 3. 2011.] Dostupný z WWW: <http://cs.wikipedia.org/wiki/Webov%C3%A1_aplikace>.

Seznam zkratek

ADO.NET	Microsoft ActiveX Data Objects .NET
AJAX	Asynchronous JavaScript and XML
ASP.NET	Active Server Pages.NET
CLR	common language runtime
CSS	Cascading Style Sheets
C#, J#, Visual Basic	programovací jazyky
DHTML	Dynamic HTML
ERD	Entity Relationship Diagram
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IL	Intermediate Language
JavaScript	skriptovací jazyk
JIT	Just In Time, speciální metoda překladu
LINQ	Language Integrated Query
MS SQL	Microsoft Structured Query Language
OOP	objektově orientované programování
PHP	Hypertext Preprocessor
UML	Unified Modeling Language
UseCase	případ užití
XHTML	eXtensible HyperText Markup Language
XML	eXtended Markup Language
.NET Framework	platforma pro osobní počítače s operačním systémem Windows
.NET	aplikační platforma společnosti Microsoft

Prohlášení o využití výsledků diplomové práce

Prohlašuji, že

- jsem byl(a) seznámen(a) s tím, že na mou diplomovou (bakalářskou) práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo;
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou (bakalářskou) práci užít (§ 35 odst. 3);
- souhlasím s tím, že diplomová (bakalářská) práce bude v elektronické podobě archivována v Ústřední knihovně VŠB-TUO a jeden výtisk bude uložen u vedoucího diplomové (bakalářské) práce. Souhlasím s tím, že bibliografické údaje o diplomové (bakalářské) práci budou zveřejněny v informačním systému VŠB-TUO;
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona;
- bylo sjednáno, že užít své dílo, diplomovou (bakalářskou) práci, nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Ostravě dne ……

……..

jméno a příjmení studenta

Adresa trvalého pobytu studenta:

Bratří Hlaviců 95
755 01 Vsetín

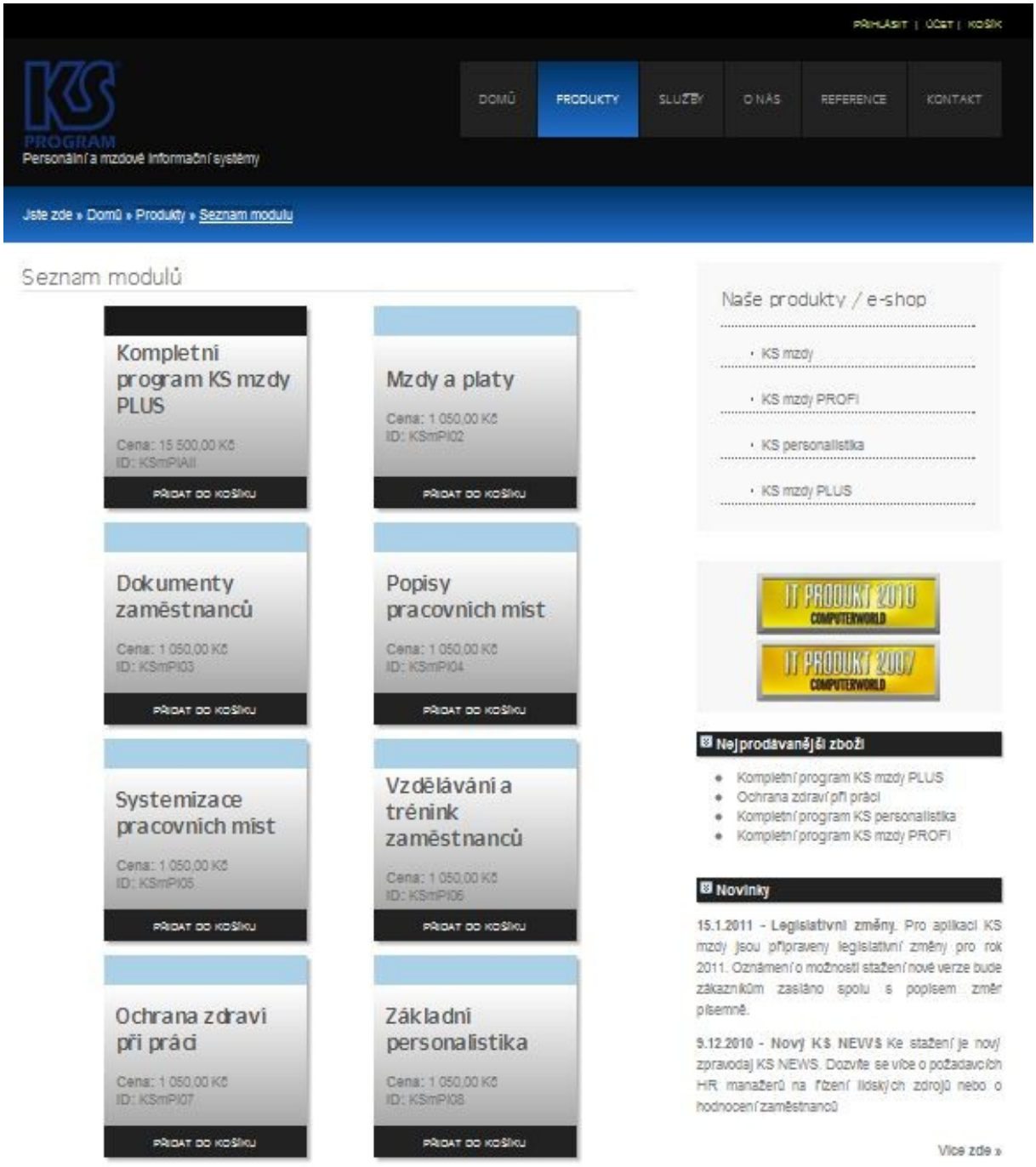
Seznam příloh

Příloha A

Vzhled webové aplikace – úvodní stránka

Obrázek A.1 Vzhled webu - úvodní stránka
zdroj: Martin Hatlapatka – vytvořeno v rámci diplomové práce

Vzhled webové aplikace – stránka s nabídkou modulů produktu KS mzdy PLUS



Obrázek A.2 vzhled webu - stránka s moduly
zdroj: Martin Hatlapatka – vytvořeno v rámci diplomové práce